

PUBLIC
2021-09-23

SAP BTP, Neo Environment

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

Content

1 What Is SAP BTP, Neo Environment. 9

1.1 Account Model. 10

Global Accounts. 10

Subaccounts. 11

Directories [Feature Set B]. 12

Relationship Between Global Accounts and Subaccounts. 13

Relationship Between Global Accounts, Subaccounts, and Directories [Feature Set B]. 14

Custom Properties [Feature Set B]. 14

User Accounts. 16

1.2 Regions in the Neo Environment. 16

Regions and Hosts Available for the Neo Environment. 17

Resilience, High Availability, and Disaster Recovery. 19

1.3 Solutions and Services. 20

1.4 Connectivity for the Neo Environment. 21

What Is Connectivity for the Neo Environment?. 26

What's New for Connectivity. 30

Administration. 54

Development. 134

Cloud Connector (Neo Environment). 232

Connectivity via Reverse Proxy. 543

Connectivity Support. 544

1.5 What is Document Service. 547

Consuming the Document Service (Java). 550

Consume Document Service (HTML5 Applications). 598

Managing Repositories in the Cockpit. 599

Managing a Repository with Console Client Commands. 602

Terminate an Account and Export Data. 603

1.6 SAP Feedback service (Beta). 603

Consuming the SAP Feedback service. 605

Getting Feedback for Applications. 615

Data Protection and Privacy. 620

1.7 Gamification. 621

Getting Started. 622

Gamification Development Cycle. 632

Gamification Workbench. 634

Security. 638

Managing Apps. . . . 641
Configuring Game Mechanics . . . 646
Integrating Gamification into a Target Application. . . . 681
Analyzing Gamification Concepts. . . . 689
Case Study: Gamified Help Desk Application. . . . 691
1.8 SAP Git Service. . . . 710
Managing Repositories. . . . 712
Working with Repositories. . . . 716
Security. . . . 720
Best Practices. . . . 723
1.9 What Is SAP Monitoring Service?. . . . 724
1.10 What's New for SAP Monitoring Service. . . . 725
2020 What's New for SAP Monitoring Service (Archive). . . . 727
2019 What's New for SAP Monitoring Service (Archive). . . . 727
2018 What's New for SAP Monitoring Service (Archive). . . . 728
1.11 Initial Setup. . . . 729
1.12 Monitoring Java Applications. . . . 729
View Monitoring Metrics of a Java Application. . . . 730
Availability Checks. . . . 732
JMX Checks. . . . 738
Generate Heap and Thread Dumps. . . . 744
Metrics REST API for Java Applications. . . . 746
Checks REST API for Java Applications. . . . 751
1.13 Monitoring HTML5 Applications. . . . 753
View Metrics of Custom Checks for an HTML5 Application. . . . 754
Configure Custom Checks for HTML5 Applications. . . . 755
Metrics REST API for HTML5 Applications. . . . 758
Checks REST API for HTML5 Applications. . . . 759
1.14 Monitoring Database Systems. . . . 760
View Monitoring Metrics of a Database System. . . . 761
View Monitoring Metrics of an SAP HANA XS Application. . . . 766
Availability Checks. . . . 767
Custom HTTP Checks. . . . 772
Metrics REST API for Database Systems. . . . 775
Metrics REST API for SAP HANA XS. . . . 776
Checks REST API for SAP HANA XS Applications. . . . 777
1.15 Alerting. . . . 779
Alert Webhook. . . . 780
Alert Email. . . . 783
1.16 Auditing and Logging Information. . . . 784
1.17 Monitoring and Troubleshooting. . . . 786

Getting Support. 786
1.18 Remote Data Sync Service. 786
Get Access to the Remote Data Sync Service. 789
Provide a MobiLink Server in Your Subaccount. 790
Develop Client-Initiated Synchronization. 792
MobiLink Server Logs. 799
Configure Authentication for a MobiLink Server. 802
Connecting SQL Anywhere Tools to MobiLink Servers. 803
Monitor the Availability of the MobiLink Server. 806
Performance and Scalability of the MobiLink Server. 807
1.19 Tutorials. 809
1.20 Tools. 811
SAP BTP Cockpit. 812
SAP Web IDE. 815
Maven Plugin. 815
SAP BTP SDK for Neo Environment. 816
SAP BTP SDK for iOS. 817
SAP JVM Tools in Eclipse for the Neo Environment. 818
Console Client for the Neo Environment. 818
Working with Cloud Management Tools Feature Set B in the Neo Environment. 819

2 Get an Enterprise Account. 823
2.1 Purchase a Customer Account. 823
2.2 Join the Partner Program. 823
2.3 Commercial Models. 824
What Is the Consumption-Based Commercial Model?. 825
What Is the Subscription-Based Commercial Model?. 827

3 Getting Started, Neo Environment. 829
3.1 Getting Started with a Customer Account in the Neo Environment. 829
3.2 Getting Started with Business Applications Subscriptions in the Neo Environment. 833
Providing Multitenant Applications to Consumers in the Neo Environment. 835
Subscribe to Java Multitenant Applications in the Neo Environment. 838
Subscribe to HTML5 Multitenant Applications in the Neo Environment. 840

4 Development, Neo Environment. 842
4.1 Deploy Business Applications in the Neo Environment. 842
4.2 Developing Java in the Neo Environment. 843
Java: Getting Started. 844
Runtime for Java. 854
Development Environment. 868
Developing Java Applications. 871

Deploying and Updating Applications. 875
Debugging Applications. 895
Developing Multitenant Applications in the Neo Environment. 898
Programming with JPA and JDBC. 916
4.3 Developing SAP HANA in the Neo Environment. 983
SAP HANA: Getting Started. 983
Launch SAP HANA XS Applications. 998
Using an SAP HANA XS Database System. 999
4.4 Multitarget Applications for the Neo Environment. 1016
Create a Hello World Multitarget Application. 1017
Defining MTA Development Descriptors. 1021
Defining MTA Deployment Descriptors for the Neo Environment. 1021
Defining Multitarget Application Archives. 1023
Defining MTA Extension Descriptors. 1025
MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment 1027
Integration with Transport Management Tools. 1056
Modeling Solutions. 1062
Operating Solutions. 1089
4.5 Developing HTML5 Applications in the Neo Environment. 1117
HTML5: Getting Started. 1119
Developer's Guide. 1124
4.6 API Documentation. 1147
Using Platform APIs. 1149
4.7 Using Services in the Neo Environment. 1152
About Services in the Neo Environment. 1152
Enable Services in the Neo Environment. 1153
Disable Services in the Neo Environment. 1154

5 Extensions, Neo Environment. 1155
5.1 Extending SAP S/4HANA Cloud in the Neo Environment. 1157
Configuring Single Sign-On Between SAP S/4HANA Cloud and SAP BTP, Neo Environment 1158
Configuring the Extension Application. 1161
Configuring SAP Web IDE. 1170
5.2 Extending SAP SuccessFactors in the Neo Environment. 1175
Configuring the Subaccount in SAP BTP for SAP SuccessFactors. 1177
Installing and Configuring Extension Applications. 1189
5.3 Extending SAP Cloud for Customer in the Neo Environment. 1219
Configuring Single Sign-On. 1221
Configuring the SAP BTP Application Connectivity to SAP Cloud for Customer APIs. 1234
5.4 Extending SAP Ariba in the Neo Environment. 1240

Configure Single Sign-On (SSO). . . . 1241
Connect the Application Running on SAP BTP to SAP Ariba. . . . 1244

6 Administration and Operations, Neo Environment. . . . 1259
6.1 Account Administration in the Cockpit. . . . 1259
Navigate in the Cockpit. . . . 1259
Managing Global Accounts Using the Cockpit. . . . 1261
Managing Directories Using the Cockpit [Feature Set B]. . . . 1277
Managing Subaccounts Using the Cockpit. . . . 1282
Managing Entitlements and Quotas Using the Cockpit. . . . 1291
Managing Members Using the Cockpit. . . . 1294
6.2 Account Administration Using the SAP BTP Command Line Interface (btp CLI) [Feature Set B] . . . 1317
Download and Start Using the btp CLI Client. . . . 1319
Command Syntax of the btp CLI. . . . 1323
General Commands and Options in the btp CLI. . . . 1327
Commands in the btp CLI. . . . 1343
6.3 Account Administration in the Neo Console Client. . . . 1346
Create Subaccounts Using the Console Client in the Neo Environment. . . . 1346
Add Quotas to Subaccounts Using the Console Client. . . . 1347
Console Client for the Neo Environment. . . . 1348
6.4 Application Operations in the Neo Environment. . . . 1587
Using Multiple Subaccounts for Staged Application Development. . . . 1587
Java: Application Operations. . . . 1589
SAP HANA: Application Operations. . . . 1631
HTML5: Application Operations. . . . 1631
Configuring Application URLs. . . . 1644
Configuring Application Access via On-Premise Reverse Proxy. . . . 1668

7 Security, Neo Environment. . . . 1671
7.1 Authorization and Trust Management in the Neo Environment. . . . 1672
Securing Java Applications. . . . 1676
Securing HTML5 Applications. . . . 1744
Securing SAP HANA Applications. . . . 1746
7.2 Platform Identity Provider. . . . 1746
Overview. . . . 1746
1. Create Trust with the Identity Authentication Tenant . . . 1747
2. Add Identity Authentication Tenant Users as Subaccount Members. . . . 1749
(Optional) 3. Configure the Identity Authentication Tenant for the Required Scenarios. . . . 1750
Accessing the SAP BTP Cockpit with the Tenant User Base. . . . 1750
Using the Console Client with the Tenant User Base. . . . 1751
7.3 OAuth 2.0 Service. . . . 1751

OAuth 2.0 Authorization Code Grant. 1753
OAuth 2.0 Client Credentials Grant. 1759
OAuth 2.0 Configuration. 1760
Best Practices for Resilient OAuth 2.0 Communication. 1768
Principal Propagation to OAuth-Protected Applications. 1770
Auditing and Logging Information for OAuth 2.0 Service. 1775
7.4 Keystore Service. 1779
Keystore API. 1779
Keys and Certificates. 1781
Enabling Client Certificate Authentication. 1797
Enable Strong Encryption in Applications. 1807
Storing Passwords. 1808
7.5 Audit Logging in the Neo Environment. 1810
Audit Log Retrieval API Usage for the Neo Environment. 1810
Audit Log Retention API Usage for the Neo Environment. 1815
7.6 Principal Propagation. 1816
Principal Propagation Between Neo Subaccounts. 1817
Principal Propagation from the Neo to the Cloud Foundry Environment. 1821
Principal Propagation from the Cloud Foundry to the Neo Environment. 1827
7.7 Protection from Web Attacks. 1835
Protection from Cross-Site Scripting (XSS). 1836
Protection from Cross-Site Request Forgery. 1840
7.8 Data Protection and Privacy. 1848
Glossary for Data Protection and Privacy. 1850
Change Logging and Read-Access Logging. 1851
Personal Data Record. 1852
Deletion. 1852
Consent. 1858

8 Getting Support, Neo Environment. 1859
Procedure. 1859
1. Log On to SAP ONE Support Launchpad. 1859
2. (Optional) Search in the Knowledge Base for Existing Solutions. 1860
3. Choose Report an Incident. 1860
4. Provide System Information. 1861
5. Provide Description. 1862
6. Provide Attachments. 1863
7. Set Priority. 1864
8. Choose Support Channel. 1864
8.1 Providing Details for Database Problems in the Neo Environment. 1865
8.2 Gather Support Information. 1866
8.3 Platform Updates and Notifications in the Neo Environment. 1866

8.4 Operating Model in the Neo Environment. 1867
8.5 Support Components. 1874

1 What Is SAP BTP, Neo Environment

SAP BTP, Neo environment is an enterprise platform-as-a-service (enterprise PaaS) that provides comprehensive application development services and capabilities, which lets you build, extend, and integrate business applications in the cloud.

> → Tip
>
> **This documentation refers to SAP Business Technology Platform, Neo environment. If you are looking for documentation about other environments, see SAP Business Technology Platform.**

At its core, SAP BTP enables you to do the following:

- Extend on-premise and cloud solutions
  Personalize and extend your existing applications, regardless of whether they run on-premise or in the cloud, to achieve agility and go to market faster.
- Integrate on-premise and cloud solutions
  Bring together disparate systems and IT landscapes to streamline your business processes and increase efficiency.
- Develop new cloud applications
  Create new cloud applications with a small capital investment in software and hardware.

The Neo environment lets you develop HTML5, Java, and SAP HANA extended application services (SAP HANA XS) applications. You can also use the UI Development Toolkit for HTML5 (SAPUI5) to develop rich user interfaces for modern web-based business applications.

The Neo environment also allows you to deploy solutions on SAP BTP. In the context of SAP BTP, a solution is made up of various application types and configurations created with different technologies, designed to implement a certain scenario or task flow. You can deploy solutions by using the Change and Transport System (CTS+) tool, the console client, or the SAP BTP cockpit, which also lets you monitor your solutions. The SAP multitarget application (MTA) model encompasses and describes application modules, dependencies, and interfaces in an approach that facilitates validation, orchestration, maintenance, and automation of the application throughout its life cycle.

The Neo environment lets you use virtual machines, allowing you to install and maintain your own applications in scenarios that aren't covered by the platform.

Capabilities and Services

According to your use cases, you may want to consume a set of services that are provided by SAP BTP. For more information, see Solutions and Services [page 20].

Regions

You can deploy applications developed in the Neo environment to various SAP data centers around the world. For more information about regional availability of the Neo environment, see Regions and Hosts Available for the Neo Environment [page 17].

Integration with SAP and Non-SAP Software

SAP BTP facilitates secure integration with on-premise systems that are running software from SAP and other vendors. Using platform services such as the Connectivity service applications can establish secure connections to on-premise solutions, enabling integration scenarios with your cloud-based applications. For more information about the Connectivity service, see What Is Connectivity for the Neo Environment? [page 26].

Secure Data

The comprehensive, multilevel security measures that are built into SAP BTP are engineered to protect your mission-critical business data and assets, and to provide the necessary industry-standard compliance certifications.

Quality Certificates

Third-party certification bodies provide independent confirmation that SAP meets the requirements of international standards. You can find all certificates at https://www.sap.com/corporate/en/company/quality.html.

1.1 Account Model

Learn more about the different types of accounts on SAP BTP and how they relate to each other.

1.1.1 Global Accounts

Global accounts are hosted environments that represent the scope of the functionality and the level of support based on a customer or partner's entitlement to platform resources and services.

The global account is the realization of the commercial contract with SAP. A global account can contain one or more subaccounts in which you deploy applications, use services, and manage your subscriptions.

Global accounts are region- and environment-independent. Within a global account, you manage all of your subaccounts, which in turn are specific to regions and environments.

1.1.2 Subaccounts

Subaccounts let you structure a global account according to your organization's and project's requirements with regard to members, authorizations, and quotas.

Subaccounts in a global account are independent from each other. This is important to consider with respect to security, member management, data management, data migration, integration, and so on, when you plan your landscape and overall architecture. Each subaccount is associated with a region, which is the physical location where applications, data, or services are hosted. It is also associated with one environment. The specific region and environment are relevant when you deploy applications and access the SAP BTP cockpit. The quotas that have been purchased for a global account have to be assigned to the individual subaccounts.

1.1.2.1 Using Beta Features with Subaccounts

SAP may offer, and a customer may choose to accept access to functionality, such as a service or application, which is not generally available and has not been validated and quality assured in accordance with SAP standard processes. Such functionality is defined as a beta feature.

Beta features let customers, developers, and partners test new features on SAP BTP. The beta features have the following characteristics:

- SAP may require that customers accept additional terms to use beta features.
- Beta features are released for enterprise accounts, trial accounts, or both.
- To allow the use of beta services and applications in the subaccounts available to you in the SAP BTP cockpit, you need to set the *Enable beta features* option. You do this on global account level by choosing the edit icon on the subaccount's tile.
- No personal data may be processed by beta functionality in the context of contractual data processing without additional written agreement.

> ⚠ Caution
>
> You shouldn't use SAP BTP beta features in subaccounts that belong to productive enterprise accounts. For more information, see Important Disclaimers and Legal Information.

Related Information

Change Subaccount Details [page 1286]

1.1.2.2 Using Experimental Features in Subaccounts

SAP may choose to experiment with a feature before it decides whether to make it available for productive use. In such a case, we ask customers, developers, and partners to provide feedback on that feature.

The experimental features have the following characteristics:

- Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means, that experimental features may be changed by SAP at any time for any reason without notice.
- Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate, or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up

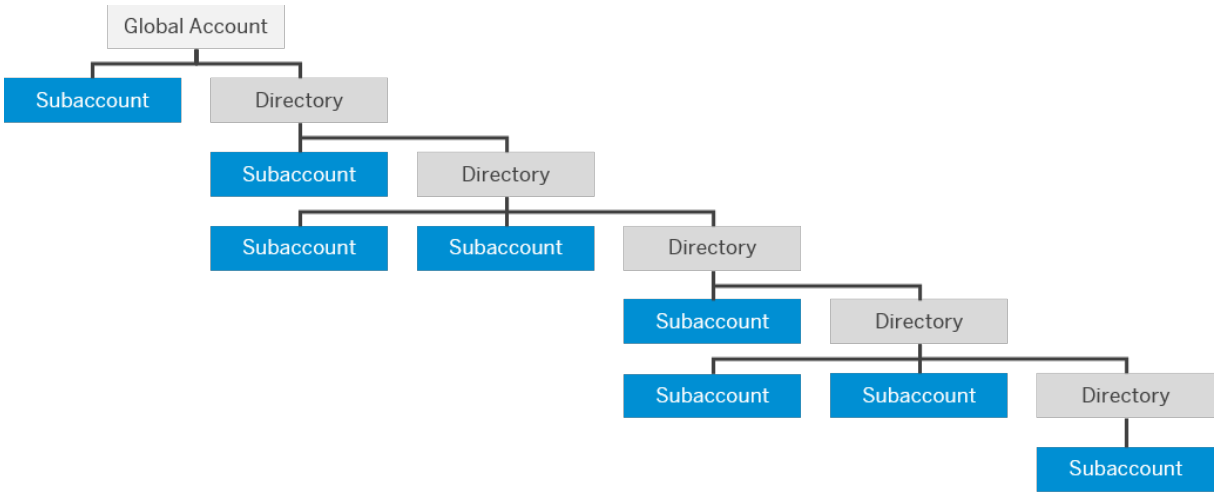
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (for example, in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

1.1.3 Directories [Feature Set B]

Directories allow you to organize and manage your subaccounts according to your technical and business needs.

A directory can contain directories and subaccounts to create a hierarchy. Using directories to group other directories and subaccounts is optional - you can still create subaccounts directly under your global account.

You can create a hierarchical structure that is 7 levels deep. The highest level of a given path is always the global account and the lowest is a subaccount, which means that you can have up to 5 levels of directories.



Directories allow you to:

- Group and filter directories and subaccounts
- Monitor usage and, for contracts that use the consumption-based model, cost
- Set custom properties and tags for identification and reporting purposes

In addition, you can also add the following features to your directories (optional):

- Manage Entitlements: Enables the assignment of a quota for services and applications to the directory from the global account quota for distribution to the directory's subaccounts.
  When you assign entitlements to a directory, you express the entitlements and maximum quota that can be distributed across its children subaccounts. You also have the option to choose the auto-assignment of a set amount of quota to all subaccounts created or moved to that directory. Subaccounts that are already in the directory when you select that option will not be auto-assigned quota.
- Manage Authorizations: Enables a custom identity provider and/or authorization management for the directory. For example, it allows certain users to manage directory entitlements. You can only use this feature in combination with the *Manage Entitlements* feature.

Related Information

Getting a Global Account
Setting Up Your Account Model
Managing Global Accounts Using the Cockpit
Managing Directories Using the Cockpit [Feature Set B] [page 1277]
Managing Subaccounts Using the Cockpit [page 1282]
Working with Global Accounts, Directories, and Subaccounts Using the btp CLI [page 1343]
Manage the Account Explorer Hierarchy [Feature Set B] [page 1279]

1.1.4 Relationship Between Global Accounts and Subaccounts

A global account can group together different subaccounts that an administrator makes available to users. Administrators can assign the available quotas of a global account to its different subaccounts and move it between subaccounts that belong to the same global account.

Subaccounts in a global account are independent from each other. This is important to consider with respect to security, member management, data management, data migration and management, integration, and so on, when you plan your landscape and overall architecture.

Each subaccount is associated with a particular region, which is the physical location where applications, data, or services are hosted. The specific region associated with a subaccount is relevant when you deploy applications (region host) and access the SAP BTP cockpit (cockpit URL). The region assigned to your subaccount doesn't have to be directly related to your location. You could be located in the United States, for example, but operate your subaccount in Europe.

For more information about the relationship between a global account and its subaccounts, see the graphic in Basic Platform Concepts. For best practices, see Setting Up Your Account Model.

You can enable a subaccount to use beta features, including services and applications, which are occasionally made available by SAP for SAP BTP. This option, unselected by default, is available only to administrators, for your enterprise account.

> ⚠ Caution
>
> You shouldn't use SAP BTP beta features in subaccounts that belong to productive enterprise accounts. For more information, see Important Disclaimers and Legal Information.

1.1.5 Relationship Between Global Accounts, Subaccounts, and Directories [Feature Set B]

A global account can group together different directories and subaccounts that an administrator makes available to users. Administrators can assign the available entitlements and quotas of a global account to its different subaccounts and move it between subaccounts that belong to the same global account.

> ℹ Note
>
> The content in this section is only relevant for cloud management tools feature set B. For more information, see Cloud Management Tools - Feature Set Overview.

The hierarchical structure of global accounts, directories, and subaccounts lets you define an account model that accurately fits your business and development needs. For example, if you want to separate development, testing, and productive usage for different departments in your organization, you can create a directory for each department, and within each directory, you group subaccounts for development, testing, and production.

1.1.6 Custom Properties [Feature Set B]

Custom properties allow you to label or tag your directories and subaccounts according to your own business and technical needs. This makes organizing and filtering your directories and subaccounts easier within your global account.

You create and assign custom properties when you create or edit a directory or subaccount. Using custom properties is optional.

Each custom property has a name (also referred to as a key) and typically one or more values that are associated with the property. You can also assign a custom property to a directory or subaccount without giving a specific value. When no value is given, the custom property behaves like a tag. Here are some examples of custom properties:

Custom Property (Name)	Property Values
Landscape	Dev Test Production

Custom Property (Name)	Property Values
Department	HR IT Finance Sales
Cost Center	000001134789 000002155534 To be defined
Flagged for Deletion	*(no values)*
Important	*(no values)*

Consider the following when working with custom properties:

- Each directory or subaccount can have up to 10 custom properties assigned to it.
- You cannot add the same custom property with a different value to the same directory or subaccount.
- Directories do not share their custom properties with subaccounts, and vice versa. However, when you view the custom properties of a subaccount, it also shows the custom properties assigned to its parent directory, if one exists.
- When you create a custom property or assign a new value to a property in a directory, they become available for use across all the directories in your global account. The also applies to subaccounts.
- Property names (keys) are case-insensitive, which means you cannot create variants of the same property name with different casing (for example, "MyKey" and "myKey" cannot coexist). Property values on the other hand are case-sensitive; be careful not to create unwanted variants of the same names or values with a different casing or styling.
- To remove a custom property that is assigned to a directory or subaccount, or to change the assigned value of a property, or to add a property, you must edit the directory or subaccount.
- Currently, you cannot manage all custom properties for all directories and subaccounts from a central page. Custom properties can only be maanged on the level of individual directories and subaccounts.

→ Tip

- You can view the custom properties that are assigned to a directory or subaccount by choosing the *More Info* option of each directory and subaccount in the *Account Explorer* page in the SAP BTP cockpit. Assigned custom properties are also listed when in the *Overview* page of every directory and subaccount.
- In the *Account Explorer* page in the cockpit, you can filter the displayed directories and subaccounts by their assigned custom properties.

Related Information

Create a Subaccount
Change Subaccount Details
Create a Directory [Feature Set B] [page 1277]
Cloud Management Tools — Feature Set Overview

1.1.7 User Accounts

A user account corresponds to a particular user in an identity provider, such as the SAP ID service (for example, an S-user, P-user) and consists, for example, of an SAP user ID and password.

Platform Users vs Business Users

There are two types of users on SAP BTP: platform users and business users. Platform users are the members of global accounts and subaccounts: usually developers, administrators or operators who deploy, administer, and troubleshoot applications and services. They can view a list of all global accounts and subaccounts, and access them using the cockpit.

Business users are those who use the applications that are deployed to SAP BTP. For example, users of subscribed apps or services, such as SAP Web IDE, are business users.

1.2 Regions in the Neo Environment

You can deploy applications in different regions. Each region represents a geographical location (for example, Europe, US East) where applications, data, or services are hosted.

All regions that are available for the Neo environment are exclusively provided by SAP. For an overview of all available regions for the Neo environment, see SAP Cloud Platform Regions and Service Portfolio.

Selecting a Region

A region is chosen at the subaccount level. For each subaccount, you select exactly one region and one environment. The selection of a region is dependent on many factors: For example, application performance (response time, latency) can be optimized by selecting a region close to the user. For more information, see *Selecting a Region* in Regions.

Deploying Applications in Regions

When deploying applications, consider that a subaccount is associated with a particular region and that this is independent of your own location. You may be located in the United States, for example, but operate your subaccount in a region in Europe.

To deploy an application in more than one region, execute the deployment separately for each host.

Regions and Hosts Available for the Neo Environment [page 17]
Each region represents a geographical location (for example, Europe, US East) where applications, data, or services are hosted.

Resilience, High Availability, and Disaster Recovery [page 19]
SAP has a number of processes in place to support resilience in SAP BTP, and provides different offerings so that you can support the high availability of your applications.

1.2.1 Regions and Hosts Available for the Neo Environment

Each region represents a geographical location (for example, Europe, US East) where applications, data, or services are hosted.

To find out about the regions available for multi-enviroment subaccounts, see Regions.

Regions, Hosts, and IP Ranges Available for the Neo Environment

Region (all operated by SAP)	Technical Key	Host	Cockpit Logon	IP Ranges
Australia (Sydney)	neo-ap1	ap1.hana.ondemand.com	Link	210.80.140.0/24, 157.133.96.0/23 and 130.214.148.64/29
Brazil (São Paulo)	neo-br1	br1.hana.ondemand.com	Link	157.133.246.0/24 and 130.214.96.64/29
Canada (Toronto)	neo-ca1	ca1.hana.ondemand.com	Link	157.133.54.0/24, 157.133.62.0/24 and 130.214.174.64/29
China (Shanghai)	neo-cn1	cn1.platform.sapcloud.cn	Link	157.133.192.128/25, 157.133.192.64/27, 157.133.194.0/24, and 130.214.152.64/29
Europe (Amsterdam)	neo-eu3	eu3.hana.ondemand.com	Link	157.133.141.0/24, 157.133.140.0/24 and 130.214.166.64/29
Europe (Frankfurt)	neo-eu2	eu2.hana.ondemand.com	Link	157.133.70.0/24, 157.133.205.0/24, 157.133.204.0/24, 157.133.206.0/24, 130.214.164.64/29 and 130.214.190.64/29
Europe (Rot)	neo-eu1	hana.ondemand.com eu1.hana.ondemand.com	Link	155.56.128.0/17 and 130.214.160.64/28

Region (all operated by SAP)	Technical Key	Host	Cockpit Logon	IP Ranges
Japan (Tokyo)	neo-jp1	jp1.hana.ondemand.com	Link	157.133.150.0/24 and 130.214.245.32/29
Kingdom of Saudi Arabia (Riyadh)	neo-sa1	sa1.hana.ondemand.com	Link	157.133.93.0/24 and 130.214.223.32/29
Russia (Moscow)	neo-ru1	ru1.hana.ondemand.com	Link	157.133.2.0/24 and 130.214.170.64/29
UAE (Dubai)	neo-ae1	ae1.hana.ondemand.com	Link	157.133.85.0/24 and 130.214.251.32/29
US East (Ashburn)	neo-us1	us1.hana.ondemand.com	Link	65.221.12.0/24, 206.112.73.0/24, 157.133.18.0/24, 157.133.16.0/24, 130.214.180.64/29
US West (Chandler)	neo-us2	us2.hana.ondemand.com	Link	64.95.110.0/24, 64.95.111.0/24, 157.133.24.0/24, 157.133.25.0/25, 157.133.26.0/24, 130.214.255.32/29
US East (Sterling)	neo-us3	us3.hana.ondemand.com	Link	169.145.117.0/24, 169.145.118.0/24, 169.145.125.0/25 and 130.214.178.64/29
US West (Colorado Springs)	neo-us4	us4.hana.ondemand.com	Link	157.133.45.0/24 and 130.214.182.64/29

IP Range Notation

The IP ranges listed here are displayed in the Classless Inter-Domain Routing (CIDR) notation. The CIDR notation is a compact way to specify an IP address and its associated routing prefix. The notation is made up of the IP address, a forward slash character (/), and the number of leading 1-bits in the subnet mask.

For example: The CIDR notation 157.133.246.0/24 consists of the IP ranges between 157.133.246.0 and 157.133.246.255.

For more information about how CIDR ranges represent multiple IP addresses, you can read online about CIDR notation.

1.2.2 Resilience, High Availability, and Disaster Recovery

SAP has a number of processes in place to support resilience in SAP BTP, and provides different offerings so that you can support the high availability of your applications.

How SAP Provides Resilience

SAP applies resilience principles when developing, updating, and deploying our SAP BTP applications and services. In the Neo environment, SAP provides resilience through the following:

- **Backups and Disaster Recovery**
  In the Neo environment, we back up all SAP BTP components every 24 hours. The backups are replicated to a secondary site, from where they can be restored in case of a disaster. For more information, see Standard Disaster Recovery and Disaster Recovery as Part of the Business Continuity Plan [page 19].
- **High Availability for SAP HANA**
  If you use the SAP BTP, SAP HANA service, you can set up your databases in high availability mode to increase the availability of your systems. For more information, see Managing Database Systems in a High Availability Setup.

Best Practices for Resilient Applications

In addition to the services offered by SAP BTP, you can follow our best practices for developing and deploying applications, which help you to make your applications running on SAP BTP stable and highly available:

- **Develop Resilient Applications**
  When developing your applications, apply the principles and patterns of resilient software design that fit your use case. For more information, see Developing Resilient Apps on SAP BTP.
- **Implement Failover**
  Use a multi-region setup and implement automatic failover to ensure the high availability of your applications. For more information, see Planning Failover on SAP BTP.

1.2.2.1 Disaster Recovery as Part of the Business Continuity Plan

The cloud platform disaster recovery (DR) plan is part of the overall cloud platform business continuity plan, which includes crisis management and process continuity activities that are triggered by a declared disaster.

What Constitutes a Disaster?

A disaster is declared by SAP when there is a loss of utilities and services, and uncertainty about whether they can be restored within a reasonable period of time. A disaster can be caused by a natural catastrophe or a man-

made incident. As long as the production site has power and is connected to the Internet, it's not considered a disaster.

How Is a Disaster Declared?

Emergency incidents are assessed by SAP as part of its business continuity plan and an SAP management member with proper authorization must officially declare a disaster to initiate a disaster recovery plan.

If a disaster is declared, operations are moved to a disaster recovery site based on the process laid out in the business continuity plan.

Standard Disaster Recovery

SAP can restore productive tenants from backups as soon as practicable in case of a disaster resulting in the loss of the primary production data center.

As the magnitude of a disaster is unpredictable, a region might not be restored in a reasonable time. In addition, a new infrastructure might need to be set up at a different location, which might require the purchase and setup of new hardware. Therefore, we can't guarantee any fixed recovery timelines.

1.3 Solutions and Services

Consume the solutions and services by SAP BTP according to your use cases.

Solutions

SAP BTP offers fast in-memory processing, sustainable, agile **solutions** and **services** to integrate data and extend applications, and fully embedded analytics and intelligent technologies.

Services

Services enable, facilitate, or accelerate the development of business applications and other platform services on SAP BTP. Services are grouped into the following service types:

- **Business services**: Services that enable, facilitate, or accelerate the development of business process components or provide industry-specific functionalities or content within a business application.
- **Technical services**: Services that enable, facilitate, or accelerate the development of general or domain independent content within a business application, independent of the application's business process or task.

You find all available services, solutions, and use cases in the SAP Discovery Center.

Use the service catalog to access service-specific resources:



1.4 Connectivity for the Neo Environment

SAP BTP Connectivity: overview, features, restrictions.

> **i** Note
>
> This documentation refers to SAP BTP, Neo environment. If you are looking for information about the Cloud Foundry environment, see Connectivity (Cloud Foundry environment).

Content

In this Topic

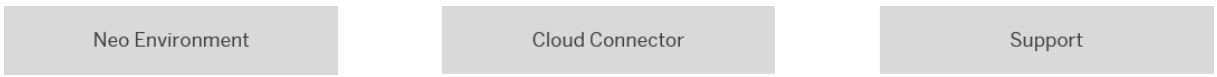
Hover over the elements for a description. Click an element for more information.



- Overview [page 22]
- Features [page 23]
- Restrictions [page 24]

In this Guide

Hover over the elements for a description. Click an element for more information.

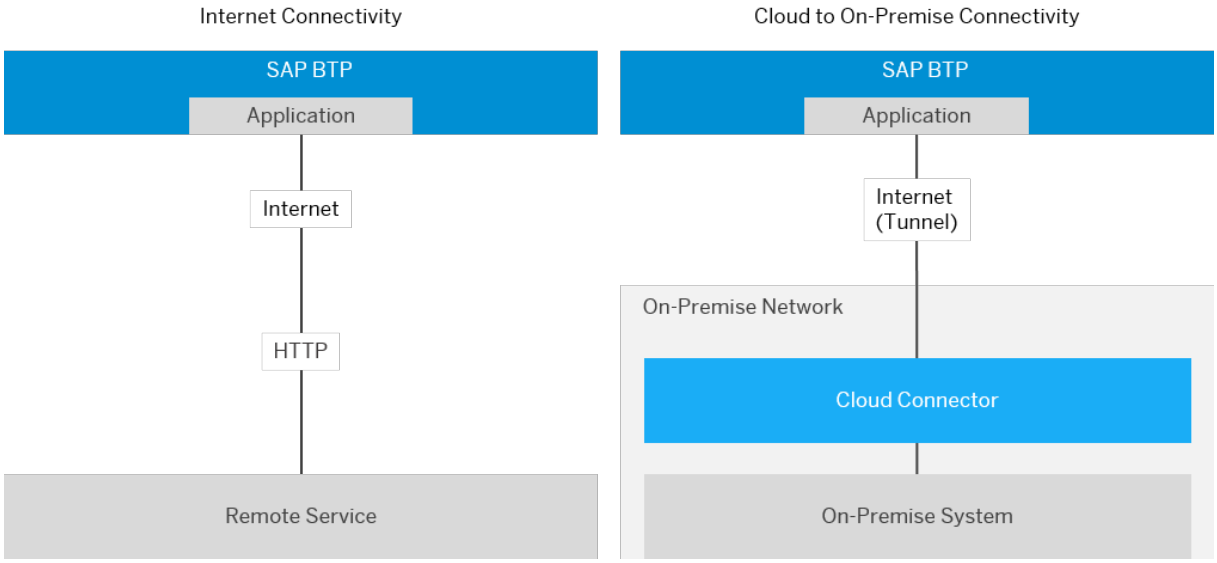


- What Is Connectivity for the Neo Environment? [page 26]
- Cloud Connector (Neo Environment) [page 232]
- Connectivity Support [page 544]

Overview

The Connectivity service allows SAP BTP applications to securely access remote services that run on the Internet or on-premise. This service:

- Allows subaccount-specific configuration of application connections via destinations.
- Provides a Java API that application developers can use to consume remote services.
- Allows you to make connections to on-premise systems, using the Cloud Connector.
- Lets you establish a secure tunnel from your on-premise network to applications on SAP BTP, while you keep full control and auditability of what is exposed to the cloud.



A typical scenario for connecting your on-premise network to SAP BTP looks like this:

- Your company owns a global account on SAP BTP and one or more subaccounts that are assigned to this global account.
- Using SAP BTP, you subscribe to or deploy your own applications.
- To connect to these applications from your on-premise network, the Cloud Connector administrator sets up a secure tunnel to your company's subaccount on SAP BTP.
- The platform ensures that the tunnel can only be used by applications that are assigned to your subaccount.
- Applications assigned to other (sub)accounts cannot access the tunnel. It is encrypted via transport layer security (TLS), which guarantees connection privacy.

For inbound connections (calling an application or service on SAP BTP from an external source), you can use Cloud Connector service channels [page 443] (on-premise connections) or the respective host [page 17] of your SAP BTP region (Internet connections).

Back to Content [page 21]

Features

The Connectivity service supports the following protocols and scenarios:

Protocol	Scenario
HTTP(S)	Exchange data between your cloud application and Internet services or on-premise systems. • Create and configure HTTP destinations to make Web connections. • Connect to on-premise systems via HTTP, using the Cloud Connector.
RFC	Invoke on-premise ABAP function modules via RFC. • Create and configure RFC destinations. • Make connections to back-end systems via RFC, using the Cloud Connector.
TCP	Access on-premise systems via TCP-based protocols using a SOCKS5 proxy.
LDAP	Enables LDAP-based user management if you are operating an LDAP server within your network.
Mail Protocols	Applications use the standard `javax.mail` API. The e-mail provider and e-mail account are configured using mail destinations. • The SMTP protocol allows you to send e-mail messages from your Web applications using e-mail providers that are accessible on the Internet, such as Google Mail (Gmail). • The IMAP and POP3 protocols let you retrieve e-mails from the mailbox of your e-mail account.

Back to Content [page 21]

Restrictions

General [page 24]

Protocols [page 25]

Neo Environment [page 25]

Cloud Connector [page 26]

> **i** Note
>
> For information about general SAP BTP restrictions, see Prerequisites and Restrictions.

General

Topic	Restriction
Java Connector	To develop a Java Connector (JCo) application for RFC communication, your SDK local runtime must be hosted by a 64-bit JVM, on a x86_64 operating system (Microsoft Windows OS, Linux OS, or Mac OS X). On Windows platforms, you must install the **Microsoft Visual Studio C++ 2013** runtime libraries (vcredist_x64.exe), see Visual C++ Redistributable Packages for Visual Studio 2013 .
Ports	For Internet connections, you are allowed to use any port >`1024`. For cloud to on-premise solutions there are no port limitations.
Destination Configuration	• You can use destination configuration files with extension `.props`, `.properties`, `.jks`, and `.txt`, as well as files with no extension. • If a destination configuration consists of a keystore or truststore, it must be stored in JKS files with a standard `.jks` extension.

Back to Restrictions [page 24]

Protocols

For the cloud to on-premise connectivity scenario, the following protocols are currently supported:

Protocol	Info
HTTP	HTTPS is not needed, since the tunnel used by the Cloud Connector is TLS-encrypted.
RFC	You can communicate with SAP systems down to SAP R/3 release 4.6C.
TCP	You can use TCP-based communication for any client that supports SOCKS5 proxies.
LDAP	Enables LDAP-based user management if you are operating an LDAP server within your network.

Back to Restrictions [page 24]

Neo Environment

Topic	Restriction
E-Mail	• You cannot communicate with an e-mail provider via an unencrypted SMTP protocol on port 25. • Fetched e-mail is not scanned for viruses. • Sending e-mail with attachments using `javax.activation.DataHandler` works with SAP BTP SDK for Java EE 6 Web Profile. • Mail destinations can be configured only on application level. That is, configuration on a subscription or customer subaccount level is not supported. • For SAP BTP SDK for Java Web and SAP BTP SDK for Java EE 6 Web Profile: Applications must use the `javax.mail` version that is provisioned by the SAP BTP runtime (see Connectivity and Destination APIs [page 135]). Applications must not include the `javax.mail` library as part of the web archive.

Back to Restrictions [page 24]

Cloud Connector

Topic	Restriction
Scenarios	To learn in which system landscapes you can set up the Cloud Connector, see Extended Scenarios [page 236].
Installation	To check all software and hardware restrictions for working with the Cloud Connector, see Prerequisites [page 239].

Back to Restrictions [page 24]

Back to Content [page 21]

Related Information

What's New for Connectivity [page 30]
Connectivity via Reverse Proxy [page 543]

1.4.1 What Is Connectivity for the Neo Environment?

Use SAP Connectivity service for your application in the Neo environment. Learn about destination management, connectivity scenarios, and required user roles.

> **i Note**
>
> This documentation refers to SAP BTP, Neo environment. If you are looking for information about the Cloud Foundry environment, see Connectivity (Cloud Foundry environment).

Content

In this Topic

Hover over the elements for a description. Click an element for more information.



- Destinations [page 27]
- User Roles [page 28]
- Scenarios [page 28]

In this Guide

Hover over the elements for a description. Click an element for more information.



- Administration [page 54]
- Development [page 134]

Destinations

To use of the Connectivity service, you must first create and configure destinations, using the corresponding communication protocol and other destination properties.

You have several options to create and edit destinations, see Managing Destinations [page 54].

To learn how to configure a destination for a specific protocol, see:

- HTTP Destinations [page 91]

- RFC Destinations [page 113]
- LDAP Destinations [page 125]
- Mail Destinations [page 127]

Back to Content [page 27]

Scenarios

Scenario	More Information
Connect Web applications and external servers via HTTP	Consume Internet Services (Java Web or Java EE 6 Web Profile) [page 155]
Make connections between Web applications and on-premise backend services via HTTP	Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170]
Connect Web applications and on-premise backend services via RFC	Invoke ABAP Function Modules in On-Premise ABAP Systems [page 194]
Use LDAP-based user authentication for your cloud application	LDAP Destinations [page 125]
Access on-premise systems via TCP-based protocols using a SOCKS5 proxy	Using the TCP Protocol for Cloud Applications [page 204]
Send and fetch e-mail via mail protocols	Sending and Fetching E-Mail [page 208]

Back to Content [page 27]

User Roles

The following user groups are involved in the end-to-end use of the Connectivity service:

- **Application operators** - are responsible for productive deployment and operation of an application on SAP BTP. They are also responsible for configuring destinations and certificates for the remote connections that an application may need, see Administration [page 54].
- **Application developers** - create a connectivity-enabled SAP BTP application by using the Connectivity service API, see Development [page 134].
- **IT administrators** - set up the connectivity to SAP BTP in your on-premise network, using the Cloud Connector [page 232].

Some procedures on the SAP BTP can be done by developers as well as by application operators. Others may include a mix of development and operation tasks. These procedures are labeled using icons for the respective task type in the corresponding task topics.

Task Types		
 Operator	 Developer	 Operator and/or Developer

To perform connectivity tasks in the Neo environment, the following user roles and authorizations apply:

Connectivity UI (Level) in the Cockpit	Subaccount Roles	OAuth Scopes
Cloud Connectors (manage)	`Administrator` `Cloud Connector Admin`	`manageSCCTunnels`
Cloud Connectors (view)	A role containing the permission `readSCCTunnels`, for example, the predefined role `Cloud Connector Admin`.	`readSCCTunnels`
Destinations (*subaccount* level)	`Administrator`	`manageSCCTunnels`
Destinations (*application* level)	`Administrator` `Developer`	`manageSCCTunnels`
Destinations (*subscription* level)	`Administrator`	`manageSCCTunnels`
Destinations (*service*)	`Administrator`	`manageSCCTunnels`

For more information on the configuration levels available for destination management, see Managing Destinations [page 54] (section *Configuration Levels (HTTP and RFC)*).

See also:

Managing Member Authorizations in the Neo Environment [page 1301]

Managing Roles [page 1710]

Platform Scopes [page 1307]

Back to Content [page 27]

Related Information

Prerequisites and Restrictions
Administration [page 54]
Development [page 134]
Security
Multitarget Applications for the Neo Environment [page 1016]
Prerequisites and Restrictions

1.4.2 What's New for Connectivity

Find the latest features, enhancements and bug fixes for SAP BTP Connectivity .

What's New for Connectivity

Related Information

2020 Connectivity (Archive) [page 30]
2019 Connectivity (Archive) [page 38]
2018 Connectivity (Archive) [page 46]
2017 Connectivity (Archive) [page 50]

1.4.2.1 2020 Connectivity (Archive)

2020

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Java Connector (JCo) - Client Certificates	JCo provides the new property `jco.client.tls_client_certificate_logon` to support the usage of a TLS client certificate for logging on to an ABAP system via WebSocket RFC. For more information, see: User Logon Properties (Cloud Foundry environment) User Logon Properties (Neo environment) For more information on WebSocket RFC, see also: WebSocket RFC	New	2020-12-17
Connectivity	Integration Suite	Cloud Foundry	HTTP Destinations - Authentication Types	Authentication type *SAP Assertion SSO* is deprecated. It will soon be removed as a feature from the Destination service. Use Principal Propagation SSO Authentication instead, which is the recommended mechanism for establishing single sign-on (SSO).	Deprecated	2020-12-17

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo	HTTP Destinations - Authentication Types	Authentication type *SAP Assertion SSO* is deprecated. Use Principal Propagation SSO Authentication instead, which is the recommended mechanism for establishing single sign-on (SSO).	Deprecated	2020-12-17
Connectivity	Integration Suite	Cloud Foundry	HTTP Destinations - Destination Properties	The destination property `SystemUser` for the authentication types: • OAuth SAML Bearer Assertion Authentication • SAP Assertion SSO Authentication will be removed soon. More information on timelines and required actions will be published in the release notes at a later stage. See also: OAuth SAML Bearer Assertion Authentication SAP Assertion SSO Authentication	Announcement	2020-12-03
Connectivity	Integration Suite	Neo Cloud Foundry	JCo Runtime - Enhancement	JCo Runtime 3.1.3.0 introduces the following enhancement: If the backend is known to be new enough, JCo does not check for the existence of RFC_METADATA_GET, thus avoiding the need to provide additional authorizations for the repository user.	New	2020-11-05
Connectivity	Integration Suite	Neo Cloud Foundry	JCo Runtime - Bug Fix	JCo Runtime 3.1.3.0 provides the following bug fix: Up to JCo 3.1.2, the initial value for fields of type STRING and XSTRING was *null*. Since the initial value check in ABAP is different, JCo now behaves the same way and uses an emtpy string and an empty byte array, respectively.	Changed	2020-11-05
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Automatic Token Retrieval	The Destination service offers a new feature related to the automatic token retrieval functionality, which lets the destination administrator define HTTP headers and query parameters as additional configuration properties, used at runtime when requesting the token service to obtain an access token. See HTTP Destinations.	New	2020-11-05

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Documentation - Principal Propagation Scenarios	The documentation of principal propagation (user propagation) scenarios provides improved information on the basic concept and guidance on how to set up different scenarios. See Principal Propagation.	Changed	2020-10-22
Connectivity	Integration Suite	Cloud Foundry	Cloud Connector 2.12.5 - Enhancements	Release of Cloud Connector version 2.12.5 introduces the following improvements: • For principal propagation scenarios, custom attributes stored in `xs.user.attributes` of the JWT (JSON Web token) are now accessible for the subject pattern. See Configure a Subject Pattern for Principal Propagation. • Improved resolving for DNS names with multiple IP addresses by adding randomness to the choice of the IP to use. This is relevant for many connectivity endpoints in SAP Cloud Platform, Cloud Foundry environment.	New	2020-10-22
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.5 - Fixes	Release of Cloud Connector version 2.12.5 provides the following bug fixes: • After actively performing a master-shadow switch for a disaster recovery subaccount, a zombie connection could cause a timeout of all application requests to on-premise systems. This issue has been fixed. • When refreshing the subaccount certificate in an high availability setup, transferring the changed certificate to the shadow was not immediately triggered, and the updated certificate could get lost. This issue has been fixed. • If many RFC connections were canceled at the same time, the Cloud Connector could crash in the native layer, causing the process to die. This issue has been fixed. • The LDAP configuration test now supports all possible configuration parameters.	Changed	2020-10-22

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Service Instances - Quota Management	When using service plan “lite”, quota management is no longer required for this service. From any subaccount you can consume the service using service instances without restrictions on the instance count. Previously, access to service plan “lite” has been granted via entitlement and quota management of the application runtime. It has now become an integral service offering of SAP Cloud Platform to simplify its usage. See also Create and Bind a Connectivity Service Instance.	Changed	2020-10-08
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Service Instances - Quota Management	When using service plan “lite”, quota management is no longer required for this service. From any subaccount you can consume the service using service instances without restrictions on the instance count. Previously, access to service plan “lite” has been granted via entitlement and quota management of the application runtime. It has now become an integral service offering of SAP Cloud Platform to simplify its usage. See also Create and Bind a Destination Service Instance.	Changed	2020-10-08
Connectivity	Integration Suite	Cloud Foundry	SAP Java Buildpack - Java Connector (JCo)	The SAP Java Buildpack has been updated from 1.27.3. to 1.28.0. • TomEE Tomcat has been updated from 7.0.104 to 7.0.105. • SAPJVM has been updated to 81.65.65. • The *com.sap.cloud.security.xsuaa* API has been updated from 2.7.5 to 2.7.6. • The SAP HANA driver has been updated from 2.5.49 to 2.5.52. • **JCo-corresponding libraries** have been updated: *connectivity* to 3.3.3, *connectivity apiext* to 0.1.37. • The **activation process for the JCo component** in the SAP Java Buildpack has been changed. Starting with this release, it is activated by setting the following environment variable: **<USE_JCO=true>**. **i Note** The **previous activation process** for the JCo component is deprecated and **will expire after a transition period**.	Changed	2020-09-24

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Error Handling	Error handling has been improved for updating service instances via the Cloud Foundry CLI and the cloud cockpit when providing the configuration JSON data.	Changed	2020-09-10
Connectivity	Integration Suite	Neo	Connectivity Service - Bug Fix	A synchronization issue has been fixed on cloud side that in very rare cases could lead to a *zombie* tunnel from the Cloud Connector to SAP Cloud Platform, which required to reconnect the Cloud Connector.	Changed	2020-09-10
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fix	During *Check Connection* processing of a destination with basic authentication, the Destination service now uses the user credentials for both the HTTP HEAD and HTTP GET requests to verify the connection on HTTP level.	Changed	2020-09-10
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fix	Using authentication type `OAuth2SAMLBearerAssertion`, an issue could occur when adding the user's SAML group attributes into the resulting SAML assertion that is sent to the target token service. This issue has been fixed.	Changed	2020-08-13
Connectivity	Integration Suite	Cloud Foundry	Destination Service REST API - Pagination Feature	The REST API pagination feature provides improved error handling in case of issues with the pagination, for example, if an invalid page number is provided.	Changed	2020-08-13
Connectivity	Integration Suite	Neo	HttpDestination Library - New Version	The `HttpDestination` v2 library has been officially released in the Maven Central Repository . It enables the usage in Tomcat and TomEE-based runtimes the same way as in the deprecated *JavaWeb* and *Java EE 6 Web Profile* runtimes. See also HttpDestination Library.	New	2020-07-30
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fix	An error handling issue has been fixed in the Destination service, which is related to the recently introduced *SAP Assertion SSO authentication* type. If a wrong input was provided, you can now see the error properly, and recover it.	Changed	2020-07-30

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Destinations - Authentication Types	You can use authentication type `OAuth2JWTBearer` when configuring a Destination. It is a simplified version of the authentication type `OAuth2UserTokenExchange` and represents the official OAuth grant type for exchanging OAuth tokens. See HTTP Destinations.	New	2020-07-02
Connectivity	Integration Suite	Cloud Foundry	Destination Service - HTTP Header	The Destination service provides a prepared HTTP header that simplifies application and service development. See HTTP Destinations (code samples).	New	2020-07-02
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Bug Fix	A concurrency issue in the Destination service, related to parallel auth token retrieval in the token cache functionality, could result in partial request failures. This issue has been fixed.	Changed	2020-07-02
Connectivity	Integration Suite	Cloud Foundry	HTTP Destinations - Authentication Types	The Cloud Foundry environment supports `SAP Assertion SSO` as authentication type for configuring destinations in the Destination service. See HTTP Destinations.	New	2020-06-18
Connectivity	Integration Suite	Cloud Foundry	Destination Service REST API	The "Find Destination" REST API now includes the scopes of the automatically retrieved access token in the response that is returned to the caller. See "Find Destination" Response Structure.	New	2020-06-04
Connectivity	Integration Suite	Cloud Foundry	Destinations for Service Instances	For subscription-based scenarios, you can use an automated procedure to create a destination that points to your service instance. See Managing Destinations.	New	2020-06-04
Connectivity	Integration Suite	Neo Cloud Foundry	Connectivity Service - Bug Fix	In rare cases, establishing a secure tunnel between Cloud Connector (version 2.12.3 or older) and the Connectivity service could cause an issue that requires to manually disconnect and connect the Cloud Connector. This issue has been fixed. The fix requires Cloud Connector version 2.12.4 or higher.	Changed	2020-05-21

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.4 - Features	Release of Cloud Connector version 2.12.4 introduces the following features and enhancements: • You can activate the SSL trace in the Cloud Connector administration UI also for the shadow instance.	New	2020-05-07
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.4 - Fixes	Release of Cloud Connector version 2.12.4 provides the following bug fixes: • You can edit and delete domain mappings in the Cloud Connector administration UI correctly. • The REST API does no longer return an empty configuration. • REST API DELETE operations do not require setting a content-type application/json to function properly. • If more than 2000 audit log entries match a selection, redefining the search and getting a shorter list now works as expected. • A potential leak of HTTP backend connections has been closed.	Changed	2020-05-07
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Bug Fix	A fix has been applied in the Connectivity service internal load balancers, enabling the sending of TCP keep-alive packets on client and server side. This change mainly affects SOCKS5-based communication scenarios.	Changed	2020-03-26
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Service Instances	You can create a service instance specifying an update policy. This allows you to avoid name conflicts with existing destinations. See Create and Bind a Destination Service Instance.	New	2020-03-26
Connectivity	Integration Suite	Cloud Foundry	Cockpit - Destination Management	The *Destinations* editor in the cockpit is available for accounts running on the cloud management tools feature set B. See Managing Destinations.	New	2020-03-12
Connectivity	Integration Suite	Neo	Connectivity Service - Bug Fix	When creating or editing a destination with authentication type `OAuth2ClientCredentials` in the cockpit, the parameter `Audience` could not be added as additional property. This issue has been fixed.	Changed	2020-03-12

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.3 - Features	Release of Cloud Connector version 2.12.3 introduces the following features and enhancements: • When using the SAP JVM as runtime, the thread dump includes additional information about currently executed RFC function modules. • The hardware monitor includes a Java Heap history, showing the usage in the last 24 hours. • If you are using the file `scc_daemon_extension.sh` to extend the daemon in a Linux installation, the content is included in the initialization section of the daemon. This lets you make custom extensions to the daemon that survive an upgrade. See Installation on Linux OS, section *Installer Scenario*.	New	2020-02-27

Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.3 - Fixes	Release of Cloud Connector version 2.12.3 provides the following bug fixes:
				• When switching roles between master and shadow instance in a high availability setup, the switch is no longer blocked by active RFC function module invocations.
				• A fix in the backend HTTP connection handling prevents issues when the backend tries to send the HTTP response before completely reading the HTTP request.
				• When sending large amounts of data to an on-premise system, and using RFC with a network that provides large bandwidth, the Cloud Connector could fail with the error message *Received invalid block with negative size*. This issue has been fixed.
				• The Cloud Connector admin UI now shows the correct user information for installed Cloud Connector instances in the *About* window.
				• Fixes in the context of disaster recovery:
				  ○ The location ID is now handled properly when setting it *after* adding the recovery subaccount.
				  ○ Application trust settings and application-specific connections are applied in the disaster case.
				  ○ Principal propagation settings are applied in the disaster case
					Changed	2020-02-27

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	JCo Runtime - WebSocket RFC	The JCo runtime in SAP Cloud Platform lets you use WebSocket RFC (RFC over Internet) with ABAP servers as of S/4HANA (on-premise) version 1909. In the RFC destination configuration, this is reflected by new configuration properties and by the option to choose between different proxy types. See Target System Configuration (Cloud Foundry environment), or Target System Configuration (Neo environment).	New	2020-02-13
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service for Trial Accounts - Bug Fix	The Connectivity service is operational again for trial accounts. A change in the Cloud Foundry Core component caused the service not be accessible by applications hosted in DiegoCell that are dedicated for trial usage in a separate VPC (virtual private cloud) account. This issue has been fixed.	Changed	2020-01-30

1.4.2.2 2019 Connectivity (Archive)

2019

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.2 - Features	Release of Cloud Connector version 2.12.2 introduces the following features and enhancements: • You can turn on the TLS trace from the Cloud Connector administration UI instead of modifying the `props.ini` file on OS level. See Troubleshooting. • The status of the used subaccount certificate is shown on the *Subaccount* overview page of the Cloud Connector administration UI, in addition to expiring certificates shown in the *Alerting* view. See Establish Connections to SAP Cloud Platform.	New	2019-12-05

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.2 - Fixes	Release of Cloud Connector version 2.12.2 provides the following bug fixes: • Subject values for certificates requiring escaping are treated correctly. • Establishing a connection to the master is now possible when being logged on to the shadow with a user that has a space in its name. • Performance statistics could show too long total execution times. This issue has been fixed. • IP address changes for the connectivity service hosts are recognized properly. • The Cloud Connector could crash on Windows, when trying to enable the payload trace with 4-eyes-principle without the required user permissions. This issue has been fixed.	Changed	2019-12-05
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Bug Fix	Applications sending a significant amount of data payload during OAuth authorization processing could cause an out-of-memory error on the Connectivity service side. This issue has been fixed.	Changed	2019-11-21
Connectivity	Integration Suite	Neo	Region Europe (Frankfurt) - Change of Connectivity Service Hosts	The following IP addresses of the Connectivity service hosts for region Europe/Frankfurt (`eu2.hana.ondemand.com`) will change on **26 October 2019**: • `connectivitynotification.eu2.hana.ondemand.com`: from 157.133.70.140 (current) to 157.133.206.143 (new) • `connectivitycertsigning.eu2.hana.ondemand.com`: from 157.133.70.132 (current) to 157.133.205.174 (new) • `connectivitytunnel.eu2.hana.ondemand.com`: from 157.133.70.141 (current) to 157.133.205.233 (new) If you have allowed the current addresses or IP ranges in your firewall rules, make sure you also include the new values **before 26 October 2019**. See also: Prerequisites: Network.	Announcement	2019-10-03

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Connection Check	Using the *Destinations* editor in the cockpit, you can check connections also for on-premise destinations. See Check the Availability of a Destination.	Changed	2019-09-26
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector - Java Runtime	The support for using Cloud Connector with Java runtime version 7 will end on December 31, 2019. Any Cloud Connector version released after that date may contain Java byte code requiring at least a JVM 8. We therefore strongly recommend that you perform **fresh** installations only with **Java 8**, and **update** existing installations running with Java 7, to Java 8 as of now. See SAP Cloud Connector – Java 7 support will phase out and Update the Java VM.	Announcement	2019-09-13
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.1 - Features	Release of Cloud Connector version 2.12.1 introduces the following features and enhancements: • Subject Alternative Names are separated from the subject definition and provide enhanced configuration options. You can configure complex values easily when creating a certificate signing request. See Exchange UI Certificates in the Administration UI. • In a high availability setup, the master instance detection no longer switches automatically if the configuration between the two instances is inconsistent. • Disaster recovery switch back to main subaccount is periodically checked (if not successful) every 6 hours. • Communication to on-premise systems supports SNI (Server Name Indication).	New	2019-08-15

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12.1 - Fixes	Release of Cloud Connector version 2.12.1 provides the following bug fixes: • The communication between master and shadow instance no longer ends up in unusable clients that show 403 results due to CSRF (Cross-Site Request Forgery) failures, which could cause undesired role switches. • When restoring a backup, the administrator password check works with all LDAP servers. • The LDAP configuration test utility properly supports secure communication. • The Refresh Subaccount Certificate dialog is no longer hanging when the refresh action fails due to some authentication or authorization issue.	Changed	2019-08-15
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Scope Attribute for OAuth-based Authentication Types	You can use the `scope` destination attribute for the OAuth-based authentication types `OAuth2ClientCredentials`, `OAuth2UserTokenExchange` and `OAuth2SAMLBearerAssertion`. This additional attribute provides flexibility on destination configuration level, letting you specify what scopes are selected when the OAuth access token is automatically retrieved by the service. See HTTP Destinations.	New	2019-08-15
Connectivity	Integration Suite	Neo	JCo Runtime for SAP Cloud Platform - Features	• Additional APIs have been added to `JCoBackgroundUnitAttributes`. See API documentation for details. • If a structure or table contains only char-like fields, new APIs let you read or modify all of them at once for the structure or the current table row. See API documentation of `JCoTable` and `JCoStructure`.	New	2019-07-18
Connectivity	Integration Suite	Neo	JCo Runtime for SAP Cloud Platform - Fixes	• qRFC and tRFC requests sent to an ABAP system by JCo can be monitored again by AIF. • Structure fields of type STRING are no longer truncated if there is a white space at the end of the field.	Changed	2019-07-18

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - JCo Multitenancy	The Connectivity service supports multitenancy for JCo applications. This feature requires a runtime environment with SAP Java Buildpack version 1.9.0 or higher. See Scenario: Multitenancy for JCo Applications (Advanced).	New	2019-06-20
Connectivity	Integration Suite	Cloud Foundry	Cloud Cockpit - Cloud Connector View	The Cloud Connector view is available also for Cloud Foundry regions. It lets you see which Cloud Connectors are connected to a subaccount.	New	2019-04-25

Technical Component	Capability	Environment	Title	Description	Type	Available as of

Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12 - Features	Release of Cloud Connector version 2.12 introduces the following features and enhancements:
				• The administration UI is now accessible not only with an administrator role, but also with a display and a support role. See Configure Named Cloud Connector Users and Use LDAP for Authentication.
				• For HTTP access control entries, you can
				  ○ allow a protocol upgrade, e.g. to WebSockets, for exposed resources. See Limit the Accessible Services for HTTP(S).
				  ○ define which host (virtual or internal) is sent in the host header. See Expose Intranet Systems, Step 8.
				• A disaster recovery subaccount in disaster recovery mode can be converted into a standard subaccount, if a disaster recovery region replaces the original region permanently. See Convert a Disaster Recovery Subaccount into a Standard Subaccount.
				• A service channel overview lets you check at a glance, which server ports are used by a Cloud Connector installation. See Service Channels: Port Overview.
				• Important subaccount configuration can be exported, and imported into another subaccount. See Copy a Subaccount Configuration.
				• An LDAP authentication configuration check lets you analyze and fix configuration issues before activating the LDAP authentication. See Use LDAP for Authentication.
				• You can use different user roles to access the Cloud Connector configuration REST APIs. See Configuration REST APIs.
				• REST APIs for shadow instance configuration have been added. See Shadow Instance Configuration.
				• You can define scenarios for resources. Such a scenario can be exported, and imported into other hosts. See Configure Accessible Resources.
					New	2019-04-25

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.12 - Fixes	Release of Cloud Connector version 2.12 provides the following bug fixes: • The SAN (subjectAlternativeName) usage in certificates can be defined in a better way and is stored correctly in the certificate. See Exchange UI Certificates in the Administration UI. • `IllegalArgumentException` does not occur any more in HTTP processing, if the backend closes a connection and data are streamed. • DNS caching is now recognized in reconnect situations if the IP of a DNS entry has changed. • SNC with load balancing now works correctly for RFC SNC-based access control entries. • A master-master situation is also recognized if, at startup of the former master instance, the new master (the former shadow instance) is not reachable. • Solution management model generation works correctly for a shadow instance. • The daemon is started properly on SLES 12 standard installations at system startup.	Changed	2019-04-25
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Authentication Types	Authentication type `OAuth2SAMLBearerAssertion` provides two different types of `Token Service URL`: • `Dedicated`: used in the context of a single tenant, or • `Common`: used in the context of multiple tenants. For type `Common`, the tenant subdomain is automatically set to the target `Token Service URL`. In addition, cloud applications can use the `x-user-token` HTTP header to propagate the user access token to the external target service at runtime. By default, the user principal is processed via the authorization HTTP header. See SAML Bearer Assertion Authentication.	New	2019-04-11
Connectivity	Integration Suite	Neo Cloud Foundry	Connectivity Service - Fix	When an on-premise system closed a connection that uses an RFC or SOCKS5 proxy, the Connectivity service kept the connection to the cloud application alive. This issue has been fixed. The connection is now always closed right after sending the response.	Changed	2019-04-11

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Cloud Foundry	Connectivity Service - Protocols	The Connectivity service supports TCP connections to on-premise systems, exposing a SOCKS5 proxy to cloud applications. This feature follows the concept of binding the credentials of a Connectivity service instance. See Using the TCP Protocol for Cloud Applications.	New	2019-03-14
Connectivity	Integration Suite	Neo	Connectivity Service - Fix	After receiving an on-premise system response with HTTP header *Connection: close*, the Connectivity service kept the HTTP connection to the cloud application alive. This issue has been fixed. The connection is now always closed right after sending the response.	Changed	2019-03-14
Connectivity	Integration Suite	Neo	Cloud Connector - Certificate Update	For the Connectivity service (Neo environment), **a new, region-specific certificate authority** (X.509 certificate) is being introduced. If you use the Cloud Connector for on-premise connections to the Neo environment, you must import the new certificate authority into your trust configuration. • After the **next month** (concrete notification will be rolled out), the current certificate authority will no longer be used to issue client certificates for Cloud Connector deployments, and only the new one will be used. The Connectivity service will still trust client certificates of Cloud Connector deployments that were already issued. • After a **three-month period** (concrete notification will be rolled out), that trust will be removed and your Cloud Connector deployment must be configured to use the new client certificates. See Update the Certificate for a Subaccount.	Announcement	2019-02-28
Connectivity	Integration Suite	Cloud Foundry	Destination Service - Authentication Types	The new authentication type `OAuth2UserTokenExchange` lets your applications use an automated exchange of user access tokens when accessing other applications or services. The feature supports single-tenant and multi-tenant scenarios. See OAuth User Token Exchange Authentication.	New	2019-02-14
Connectivity	Integration Suite	Neo	RFC - Stateful Sequences	You can make a stateful sequence of function module invocations work across several request/response cycles. See Invoking ABAP Function Modules via RFC.	Changed	2019-01-31

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration Suite	Neo Cloud Foundry	Cloud Connector 2.11.3	A security note for Cloud Connector version 2.11.3 has been issued. See SAP note 2696233.	Changed	2019-01-15
Connectivity	Integration Suite	Cloud Foundry	Protocols - RFC Communication	You can use the RFC protocol to set up communication with on-premise ABAP systems for applications in the Cloud Foundry environment. This feature requires a runtime environment with SAP Java Buildpack version 1.8.0 or higher. See Invoking ABAP Function Modules via RFC.	New	2019-01-17
Connectivity	Integration Suite	Cloud Foundry	Destinations - Renew Certificates	A button in the *Destinations* editor lets you update the validity period of an X.509 certificate. See Set up Trust Between Systems.	New	2019-01-17

1.4.2.3 2018 Connectivity (Archive)

2018

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo	Connectivity Service - Performance	A change in the SAP Cloud Platform Connectivity service improves performance of data upload (on-premise to cloud) and data download (cloud to on-premise) up to 4 times and 15-30 times respectively.	Changed	2018-12-20

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo	Connectivity Service - Resilience	The Connectivity service has a better protection against zombie connections, which improves resilience and overall availability for the cloud applications consuming it.	Changed	2018-12-20
Connectivity	Integration	Neo	Password Storage Service	A Password Storage REST API is available in the SAP API Business Hub, see Password Storage (Neo Environment).	New	2018-12-06
Connectivity	Integration	Neo	Destination Configuration Service	A Destination Configuration service REST API is available in the SAP API Business Hub.	New	2018-12-06
Connectivity	Integration	Cloud Foundry	Destination Service	A Destination service REST API is available in the SAP API Business Hub.	New	2018-12-06
Connectivity	Integration	Neo	JCo Runtime for SAP Cloud Platform - Fixes	• When using `JCoRecord.fromJSON()` for a structure parameter, the data is now always sent to the backend system. Also, you do not need to append the number of provided rows for table parameters before parsing the JSON document any more. • Depending on the configuration of certain JCo properties, an internally managed connection pool could throw a `JCoException` (error group `JCO_ERROR_RESOURCE`). In a thread waiting for a free connection from this pool, an error message then erroneously reported that the pool was exhausted . This error situation could occur if the used destination was not configured with the property `jco.destination.max_get_client_time` set to 0 and the destination's `jco.destination.peak_limit` value was set higher than the `jco.destination.pool_capacity`. This issue has been fixed.	Changed	2018-12-06

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo	JCo Runtime for SAP Cloud Platform - Features	Support of the RFC fast serialization. Depending on the exchanged parameter and data types, the performance improvements for RFC communication can reach multiple factors. See SAP note 2372888 (prerequisites) and Parameters Influencing Communication Behavior [page 122] (JCo configuration in SAP Cloud Platform).	Changed	2018-12-06
Connectivity	Integration	Neo	JCo Runtime for SAP Cloud Platform - Information	Local runtimes on Windows must install the VS 2013 redistributables for x64, instead of VS 2010.	Changed	2018-12-06
Connectivity	Integration	Neo Cloud Foundry	Cloud Connector Fixes	Release of Cloud Connector 2.11.3: • An issue in RFC communication could cause the trace entry *com.sap.scc.jni.CpicCommunicationException: no SAP ErrInfo available* when the network is slow. This issue has been fixed. • The Windows service no longer runs in *error 1067* when stopped by an administrator. • In previous releases, the connection between a shadow and a master instance occasionally failed at startup and produced an empty error message. This issue has been fixed. • The Cloud Connector does not cache Kerberos tokens in the protocol handler any more, as they are one-time tokens and cannot be reused. • For HTTP access control entries, you can configure resources containing a # character.	Changed	2018-12-06

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo Cloud Foundry	Cloud Connector Enhancements	Release of Cloud Connector 2.11.3: • If the user *sapadm* exists on a system, the installation on Linux assigns it to the *sccgroup*, which is a prerequisite for solution management integration to work properly, see Configure Solution Management Integration [page 457]. • Restoring a backup has been improved. See Configuration Backup [page 460]. • The HTTP session store size has been reduced. You can handle higher loads with a given heap size. • Cipher suite configuration has been improved. Also, there is a new security status entry for cipher suites, see Recommendations for Secure Setup [page 266].	Changed	2018-12-06
Connectivity	Integration	Neo	HTTP Destinations	The *OAuth2 Client Credentials* grant type is supported by the *Destinations* editor in the SAP Cloud Platform cockpit as well as by the client Java APIs `ConnectivityConfiguration`, `AuthenticationHeaderProvider` and `HttpDestination`, available in SAP Cloud Platform Neo runtimes. See OAuth Client Credentials Authentication.	Changed	2018-10-11
Connectivity	Integration	Cloud Foundry	User Propagation	The connectivity service supports the SaaS application subscription flow and can be declared as a dependency in the get dependencies subscription callback, also via MTA (multi-target)-bundled applications. See Consuming the Connectivity Service (Cloud Foundry Environment) and Configure Principal Propagation via User Exchange Token (Cloud Foundry Environment).	Changed	2018-09-27
Connectivity	Integration	Neo Cloud Foundry	Cloud Connector 2.11.2	Release of Cloud Connector 2.11.2 • SNC configuration now provides the value of the environment variable `SECUDIR`, which you need for the usage of the SAP Cryptographic Library (SAPCRYPTOLIB). See Initial Configuration (RFC). • On Linux, the RPM (Red Hat Package Manager) now ensures that the configuration of the interaction with the SAP Host Agent (used for the Solution Manager integration) is adjusted. See Configure Solution Management Integration. • The Cloud Connector shadow instance now provides a configuration option for the connection and request timeout that may occur during health check against the master instance. See Master and Shadow Administration.	Changed	2018-08-16

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Connectivity	Integration	Neo Cloud Foundry	Cloud Connector 2.11.2	Fixes of Cloud Connector 2.11.2 • In a high availability setup, the switch from the master instance to the shadow instance occasionally caused communication errors towards on-premise systems. This issue has now been fixed. • You can now import multiple certificates with the same subject to the trust store. Details about expiration date and issuer are displayed in the tool tip. See Set Up Trust, section *Trust Store*. • You can now configure also the MOC (Multiple Origin Composition) OData service paths as resources. • The `Location` header is now adjusted correctly according to your access control settings in case of a redirect. • Principal propagation now also works with SAML assertions that contain an empty attribute element. • SAP Cloud Platform applications occasionally got an HTTP 500 (internal server error) response when an HTTP connection was closed. The applications are now always informed properly.	Changed	2018-08-16
Connectivity	Integration	Neo	HttpDestination Library	The SAP `HttpDestination` library (available in the SDK and cloud runtime "Java EE 6 Web Profile") now creates Apache `HttpClient` instances which work with strict SNI (Server Name Indication) servers. Use cases with strict SNI configuration on the server side will no longer get the error message *Failure reason: "peer not authenticated"*, that was raised either at runtime or while performing a connection test via the SAP Cloud Platform cockpit *Destinations* editor (*Check Connection* function).	Changed	2018-08-16

1.4.2.4 2017 Connectivity (Archive)

Archived release notes for 2017 and older.

28 September 2017 - Connectivity

New

The destination service (Beta) is available in the Cloud Foundry environment. See Consuming the Destination Service.

3 August 2017 - Connectivity

Enhancement

Cloud Connector

Release of SAP Cloud Platform Cloud Connector 2.10.1.

- The URLs of HTTP requests can now be longer than 4096 bytes.
- SAP Solution Manager can be integrated with one click of a button if the host agent is installed on a Cloud Connector machine. See the *Solution Management* section in Monitoring [page 479].
- The limitation that only 100 subaccounts could be managed with the administration UI has been removed. See Managing Subaccounts [page 289].

Fix

Cloud Connector

- The regression of 2.10.0 has been fixed, as principal propagation now works for RFC.
- The cloud user store works with group names that contain a backslash (\) or a slash (/).
- Proxy challenges for NT LAN Manager (NTLM) authentication are ignored in favor of Basic authentication.
- The back-end connection monitor works when using a JVM 7 as a runtime of Cloud Connector.

25 May 2017 - Connectivity

Enhancement

Cloud Connector

Release of SAP HANA Cloud connector 2.10.0.1

- Support of connectivity to an SAP Cloud Platform Cloud Foundry environment.
- Support of direct connectivity with S/4HANA Cloud systems. You can open a Service Channel to an S/4HANA Cloud system in order to use Communication Scenarios requiring RFC communication to S/4HANA Cloud. See Configure a Service Channel for RFC [page 450].
- Support of arbitrary protocols via the possibility to configure a TCP access control entry. SAP Cloud Platform Connectivity is offering a SOCKS5 proxy, with which you can address such exposed hosts. See Using the TCP Protocol for Cloud Applications [page 204].
- Support for disaster recovery events of SAP Cloud Platform regions. For each subaccount you can configure a disaster recovery subaccount for a disaster region. In case of a disaster, the disaster recovery account can be switched active immediately using the exact same configuration. See Configure a Disaster Recovery Subaccount [page 297].
- In the access control settings you can add further constraints for RFC based communication to ABAP systems: an administrator can configure, which clients shall be exposed and can define which users should not be able to access the system via Cloud Connector. See Configure Access Control (RFC) [page 338].
- You can generate self-signed certificates for CA and system certificate so that you can setup demo scenarios with principal propagation without the need of using lengthy openSSL or keytool command sequences. See Configure a CA Certificate for Principal Propagation [page 304] and Initial Configuration (HTTP) [page 285].
- A first set of monitoring HTTP APIs have been introduced: The state of all subaccount connections, a back-end connection monitor and a performance overview monitor. See Monitoring APIs [page 487].
- On Windows platforms, Cloud Connector 2.10 now requires Visual Studio 2013 runtime libraries.

Fix

Cloud Connector

- The is no longer a bottleneck that could lengthen the processing times of requests to exposed back-end systems, after many hours under high load when using principal propagation, connection pooling, and many concurrent sessions.
- Session management is no longer terminating early active sessions in principal propagation scenarios.
- On Windows 10 hardware metering in virtualized environments shows hard disk and CPU data.

11 May 2017 - Connectivity

New

In case the remote server supports only TLS 1.2, use this property to ensure that your scenario will work. As TLS 1.2 is more secure than TLS 1.1, the default version used by HTTP destinations, consider switching to TLS 1.2.

30 March 2017 - Connectivity

Enhancement

The release of SAP Cloud Platform Cloud Connector 2.9.1 includes the following improvements:

- UI renovations based on collected customer feedback. The changes include rounding offs, fixes of wrong/odd behaviors, and adjustments of controls. For example, in some places tables were replaced by *sap.ui.table.Table* for better experience with many entries.
- You can trigger the creation of a thread dump from the *Log and Trace Files* view.
- The connection monitor graphic for idle connections was made easier to understand.

Fix

- When configuring authentication for LDAP, the alternate host settings are no longer ignored.
- The email configuration for alerts is processing correctly the user and password for access to the email server.
- Some servers used to fail to process HTTP requests when using the HTTP proxy approach (HTTP Proxy for On-Premise Connectivity [page 152]) on the SAP Cloud Platform side.
- A bottleneck was removed that could lengthen the processing times of requests to exposed back-end systems under high load when using principal propagation.
- The Cloud Connector accepts passwords that contain the '§' character when using authentication-mode password.

16 March 2017 - Connectivity

Enhancement

Update of JCo runtime for SAP Cloud Platform. See Connectivity.

Fix

A `java.lang.NullPointerException` might have occurred when using a `JCoRepository` instance in roundtrip optimization mode (will be used if the JCo property *jco.use_repository_roundtrip_optimization* was set to 1 at its creation time). The `NullPointerException` was either thrown when trying to execute a *JCoFunction* object, which has been created by such a repository instance, or even earlier when querying the meta data for a *JCoFunction*, *JCoFunctionTemplate* or a *JCoRecordMetaData* object from an AS ABAP back-end system. Only certain complex data structures and table parameter definitions were affected by this bug.

Older Release Notes

- 2016
- 2015
- 2014

- 2013

1.4.3 Administration

Manage destinations and authentication for applications in the Neo environment.

Task	Description
Managing Destinations [page 54]	Create and configure destinations. You can use destinations for outbound communication between a cloud application and a remote system.
HTTP Destinations [page 91]	Create and configure HTTP destinations.
RFC Destinations [page 113]	Create and configure RFC destinations.
LDAP Destinations [page 125]	Create and configure LDAP destinations.
Mail Destinations [page 127]	Create and configure mail destinations.
Principal Propagation [page 129]	Use principal propagation to forward the identity of cloud users to a back-end system (single sign-on).
Multitenancy in the Connectivity Service [page 131]	Manage destinations for multitenancy-enabled applications that require a connection to a remote service or on-premise application.
Configuring Backup [page 134]	Find an overview of backup procedures for your destination configurations.

1.4.3.1 Managing Destinations

Overview

Destinations are used for the outbound communication of a cloud application to a remote system and contain the required connection information. They are represented by symbolic names that are used by cloud applications to refer to a remote connection.

The Connectivity service resolves the destination at runtime based on the symbolic name provided. The result is an object that contains customer-specific configuration details, for example, the URL of the remote system or service, the authentication type, and the required credentials.

To configure a destination, you can use files with extension `.props`, `.properties`, `.jks`, and `.txt`, as well as files with no extension.

Destination Names

A destination name must be unique for the current application. It must contain only alphanumeric characters, underscores, and dashes. The maximum length is 200 characters.

Destination Types

The currently supported destination types are **HTTP**, **RFC**, **LDAP** and **Mail**.

- HTTP Destinations [page 91] - provide data communication via the HTTP protocol and are used for both Internet and on-premise connections.
- RFC Destinations [page 113] - make connections to ABAP on-premise systems via RFC protocol using the Java Connector (JCo) as API.
- LDAP Destinations [page 125] - enable LDAP-based user management if you are operating an LDAP server within your network.
- Mail Destinations [page 127] - specify an e-mail provider for sending and retrieving e-mails via SMTP, IMAP, and POP3 protocols.

Configuration Tools

To configure and use a destination to connect your cloud application, you can use one of the following tools:

- Configure Destinations from the Eclipse IDE [page 64]
- Configure Destinations from the Cockpit [page 77]
- Configure Destinations from the Console Client [page 56]

Configuration Level (HTTP and RFC)

Destinations can be simultaneously configured on three levels: application, consumer subaccount, and subscription. This means it is possible to have one and the same destination on more than one configuration level.

- *Application level* - The destination is related to an application and its relevant provider subaccount. It is, though, independent from the consumer subaccount in which the application is running.
- *Consumer subaccount level* - The destination is related to a particular subaccount.
- *Subscription level* - The destination is related to the triad `<Application, Provider Subaccount, Consumer Subaccount>`.

The runtime tries to resolve a destination in the following order: *Subscription level* → *Consumer subaccount level* → *Provider application level*.

For more information about the usage of consumer subaccount, provider subaccount, and provider application, see Configure Destinations from the Console Client [page 56].

Configuration Cache

- Destination configuration files and Java keystore (JKS) files are cached at runtime. The cache expiration time is set to a small time interval (currently around 4 minutes). This means that once you update an existing destination configuration or a JKS file, the application needs about 4 minutes until the new destination configuration is applied. To avoid this waiting time, the application can be restarted on the cloud; following the restart, the new destination configuration takes effect immediately.
- When you configure a destination for the first time, it takes effect immediately.
- If you change a mail destination, the application needs to be restarted before the new configuration becomes effective.

Examples

You can find examples in the SDK package that you previously downloaded from http://tools.hana.ondemand.com.

Open the SDK location and go to `/tools/samples/connectivity`. This folder contains a standard `template.properties` file, `weather` destination, and `weather.destinations.properties` file, which provides all the necessary properties for uploading the `weather` destination.

1.4.3.1.1 Configure Destinations from the Console Client

As an application operator, you can configure your application using SAP BTP console client. You can configure `HTTP`, `Mail`, or `RFC` destinations using a standard properties file.

The tasks listed below demonstrate how to upload, download, and delete connectivity destinations. You can perform these operations for destinations related to your own subaccount, a provider subaccount, your own application, or an application provided by another subaccount.

To use an application from another subaccount, you must be subscribed to this application through your subaccount.

> **i** Note
>
> Destination files must be encoded in **ISO 8859-1** character encoding.

Prerequisites

- You have downloaded and set up the console client. For more information, see Set Up the Console Client [page 847].
- For specific information about all connectivity restrictions, see Connectivity → section "Restrictions".

HTTP Destination Properties Files

- `Name` - the name of the destination.
- `URL` - the URL of the remote system or service.
- `Authentication` - the type of authentication against the remote system or service.

The number of mandatory property keys varies depending of the authentication type you choose. For more information about HTTP destination properties files, HTTP Destinations [page 91].

Key stores and trust stores must be stored in JKS files with a standard *.jks* extension.

If mandatory fields are missing or data is specified incorrectly, you will be prompted accordingly by the console client.

Mail Destination Properties Files

- `Name` - the name of the destination.
- `Type` - must be "MAIL" for mail destinations.
- `mail.*` - `javax.mail` properties for configuring the mail session.

For more information about mail destination properties files, see Mail Destinations [page 127].

If mandatory fields are missing or data is specified incorrectly, you will be prompted accordingly by the console client.

RFC Destination Properties Files

- `Name` - the name of the destination.
- `Type` - must be "RFC" for RFC destinations.
- `jco.client*` - JCo properties for configuring an RFC connection.
- `jco.destination*` - JCo properties for configuring the behavior of a JCo destination.

All properties except `Name` and `Type` must start with "**`jco.client.`**" or "**`jco.destination`**". For more information about RFC destination properties files, see RFC Destinations [page 113].

If mandatory fields are missing or data is specified incorrectly, you will be prompted accordingly by the console client.

Tasks

- Upload Destinations [page 58]

- Download Destinations [page 60]
- Delete Destinations [page 61]

Scenarios

- Example: Send E-Mails [page 211]
- Consume Internet Services (Java Web or Java EE 6 Web Profile) [page 155]
- Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170]
- Invoke ABAP Function Modules in On-Premise ABAP Systems [page 194]

Related Information

Examples (Console) [page 63]

1.4.3.1.1.1 Upload Destinations

Context

The procedure below explains how you can upload destination configuration properties files and certificate files. You can upload them on subaccount, application or subscribed application level.

> **i** Note
>
> Bear in mind that, by default, your destinations are configured on SAP BTP, that is the *hana.ondemand.com* landscape. If you need to specify a particular region host, you need to add the `--host` parameter, as shown in the examples. Otherwise, you can skip this parameter.

Procedure

1. Open the command prompt.
2. Navigate to the `tools` folder of the SDK location.
3. Optional: Enter **`neo help`** to display all the commands of the console client or **`neo help <command_name>`** to display the help information for a command.
4. Upload a destination.

- To upload a destination on subaccount level, use the following command:

```
neo put-destination --account <subaccount_name> --user <user_name> --
localpath <destination_or_JKS_file_localpath> --host <host>
```

- To upload a destination on application level, use the following command:

```
neo put-destination --account <subaccount_name> --user <user_name> --
application <application_name> --localpath
<destination_or_JKS_file_localpath> --host <host>
```

- To upload a destination for a subscribed application, use the following command:

```
neo put-destination --account <subaccount_name> --user <user_name> --
application <provider_subaccount>:<provider_application> --localpath
<destination_or_JKS_file_localpath> --host <host>
```

Tips

> **i Note**
>
> When uploading a destination configuration file that contains a password field, the password value remains available in the file. However, if you later download this file, using the `get-destination` command, the password value will no more be visible. Instead, after `Password =...`, you will only see an empty space.

> **i Note**
>
> The configuration parameters used by SAP BTP console client can be defined in a properties file as well. This may be done instead of specifying them directly in the command (with the exception of the `–password` parameter, which must be specified when the command is executed). When you use a properties file, enter the path to it as the last command line parameter.
>
> **Example:**
>
> ```
> neo put-destination <path_to_properties_file>
> ```

Related Information

Examples (Console) [page 63]
put-destination [page 1517]

1.4.3.1.1.2 Download Destinations

Context

The procedure below explains how you can download (read) destination configuration properties files and certificate files. You can download them on subaccount, application or subscribed application level.

You can read destination files with extension **`.props`**, **`.properties`**, **`.jks`**, and **`.txt`**, as well as files with no extension. Destination files must be encoded in **ISO 8859-1** character encoding.

> **i** Note
>
> Bear in mind that, by default, your destinations are configured on SAP BTP, that is the *hana.ondemand.com* landscape. If you need to specify a particular region host, you need to add the `--host` parameter, as shown in the examples. Otherwise, you can skip this parameter.

Procedure

1. Open the command prompt.
2. Navigate to the `tools` folder of the SDK location.
3. Optional: Enter **`neo help`** to display all the commands of the console client or **`neo help <command_name>`** to display the help information for a command.
4. Download a destination.
   - To read a destination on subaccount level, use the following command:

     ```
     neo get-destination --account <subaccount_name> --user <user_name> --name
     <destination_name> --localpath <localpath_to_destination_or_JKS_file> --
     host <host>
     ```

   - To read a destination on application level, use the following command:

     ```
     neo get-destination --account <subaccount_name> --user <user_name> --
     application <application_name> --name <destination_name> --localpath
     <localpath_to_destination_or_JKS_file> --host <host>
     ```

   - To read a destination for a subscribed application, use the following command:

     ```
     neo get-destination --account <subaccount_name> --user <user_name> --
     application <provider_subaccount>:<provider_application> --name
     <destination_name> --localpath <localpath_to_destination_or_JKS_file> --
     host <host>
     ```

Tips

> **i** Note
>
> If you download a destination configuration file that contains a password field, the password value will not be visible. Instead, after `Password =...`, you will only see an empty space. You will need to learn the password in other ways.

> **i** Note
>
> The configuration parameters used by SAP BTP console client can be defined in a properties file as well. This may be done instead of specifying them directly in the command (with the exception of the `–password` parameter, which must be specified when the command is executed). When you use a properties file, enter the path to it as the last command line parameter. A sample *weather* properties file can be found in directory `<SDK_location>\tools\samples\connectivity`.
>
> **Example:**
>
> ```
> neo get-destination <path_to_properties_file>
> ```

Related Information

Examples (Console) [page 63]
put-destination [page 1517]

1.4.3.1.1.3 Delete Destinations

Context

The procedure below explains how you can delete destination configuration properties files and certificate files. You can delete them on subaccount, application or subscribed application level.

> **i** Note
>
> Bear in mind that, by default, your destinations are configured on SAP BTP, that is the *hana.ondemand.com* landscape. If you need to specify a particular region host, you need to add the `--host` parameter, as shown in the examples. Otherwise, you can skip this parameter.

Procedure

1. Open the command prompt.
2. Navigate to the `tools` folder of the SDK location.
3. Optional: Enter **`neo help`** to display all the commands of the console client or **`neo help <command_name>`** to display the help information for a command.
4. Delete a destination.
   - To delete a destination on subaccount level, use the following command:

     ```
     neo delete-destination --account <subaccount_name> --user <user_name> --
     name <destination_name> --host <host>
     ```

   - To delete a destination on application level, use the following command:

     ```
     neo delete-destination --account <subaccount_name> --user <user_name> --
     application <application_name> --name <destination_name> --host <host>
     ```

   - To delete a destination from a subscribed application, use the following command:

     ```
     neo delete-destination --account <subaccount_name> --user <user_name> --
     application <provider_subaccount>:<provider_application> --name
     <destination_name> --host <host>
     ```

Tips

> **i** Note
>
> The configuration parameters used by SAP BTP console client can be defined in a properties file as well. This may be done instead of specifying them directly in the command (with the exception of the `–password` parameter, which must be specified when the command is executed). When you use a properties file, enter the path to it as the last command line parameter.
>
> **Example:**
>
> ```
> neo delete-destination <path_to_properties_file>
> ```

Related Information

Examples (Console) [page 63]
delete-destination [page 1406]

1.4.3.1.1.4 Examples (Console)

Examples for Uploading Destinations

```
neo put-destination --account mysubaccount --user p1234567890 --localpath C:
\myfiles\myconfiguration.jks --host hanatrial.ondemand.com
```

```
neo put-destination --account mysubaccount --user p1234567890 --application demo
--localpath C:\SDK\tools\samples\connectivity\weather --host
hanatrial.ondemand.com
```

```
neo put-destination --account mysubaccount --user p1234567890 --application
<provider_account>:<provider_app> --localpath C:\SDK\tools\samples\connectivity
\weather --host hanatrial.ondemand.com
```

Examples for Downloading Destinations

```
neo get-destination --account mysubaccount --user p1234567890 --name weather --
localpath C:\myfiles --host hanatrial.ondemand.com
```

```
neo get-destination --account mysubaccount -user p1234567890 --application demo
--name myconfiguration.jks --localpath C:\SDK\tools\samples\connectivity --host
hanatrial.ondemand.com
```

```
neo get-destination --account mysubaccount --user p1234567890 --application
<provider_account>:<provider_app> --name weather --localpath C:\SDK\tools\samples
\connectivity --host hanatrial.ondemand.com
```

Examples for Deleting Destinations

```
neo delete-destination --account mysubaccount --user p1234567890 --name
myconfiguration.jks --host hanatrial.ondemand.com
```

```
neo delete-destination --account mysubaccount --user p1234567890 --application
demo --name weather --host hanatrial.ondemand.com
```

```
neo delete-destination --account mysubaccount --user p1234567890 --application
<provider_account>:<provider_app> --name weather --host hanatrial.ondemand.com
```

1.4.3.1.2 Configure Destinations from the Eclipse IDE

You can use the *Connectivity* editor in the Eclipse IDE to configure `HTTP`, `Mail`, `RFC` and `LDAP` destinations in order to:

- Connect your cloud application to the Internet or make it consume an on-premise back-end system via HTTP(S);
- Send an e-mail from a simple Web application using an e-mail provider that is accessible on the Internet;
- Make your cloud application invoke a function module in an on-premise ABAP system via RFC.
- Use LDAP-based user authentication for your cloud application.

You can create, delete and modify destinations to use them for direct connections or export them for further usage. You can also import destinations from existing files.

> **i** Note
>
> Destination files must be encoded in **ISO 8859-1** character encoding.

Prerequisites

- You have downloaded and set up your Eclipse IDE. For more information, see Setting Up the Development Environment [page 845] or Updating SAP BTP SDK for Neo Environment [page 849].
- You have created a Java EE application. For more information, see or .

Tasks

- Create and Delete Destinations Locally [page 65]
- Create and Delete Destinations on the Cloud [page 69]
- Use Destination Certificates (IDE) [page 70]
- Import Destinations (IDE) [page 72]
- Export Destinations (IDE) [page 73]

Scenarios

- Example: Send E-Mails [page 211]
- Consume Internet Services (Java Web or Java EE 6 Web Profile) [page 155]
- Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170]
- Invoke ABAP Function Modules in On-Premise ABAP Systems [page 194]

Related Information

Examples (IDE) [page 74]

1.4.3.1.2.1 Create and Delete Destinations Locally

Context

The procedure below demonstrates how you can create and configure connectivity destinations (`HTTP`, `Mail`, or `RFC`) on a local SAP BTP server.

Procedure

1. In the context menu of the *Servers* view, choose *New* > *Server* .
2. Expand the *SAP* node and, as a server type, choose between:
   - **`Java Web Server`**
   - **`Java Web Tomcat 7 Server`**
   - **`Java Web Tomcat 8 Server`**
   - **`Java EE 6 Web Profile Server`**
3. Choose *Finish*.
4. A new server `<name> [Stopped]>` appears on the *Servers* view.

   Also, a *Servers* folder is created and appears in the navigation tree of the Eclipse IDE. It contains configurable folders and files you can use, for example, to change your HTTP or JMX port.
5. On the *Servers* view, double-click the added server to open its editor.
6. Go to the *Connectivity* tab view.
   a. In the *All Destinations* section, choose the button to create a new destination.
   b. From the dialog window, enter a name for your destination, select its type and then choose *OK*.
   c. In the *URL* field, enter the URL of the target service to which the destination should refer.
   d. In the *Authentication* dropdown box, choose the authentication type required by the target service to authenticate the calls.
      - If the target service does not require authentication, choose *NoAuthentication*.
      - If the target service requires basic authentication, choose *BasicAuthentication*. You need to enter a user name and a password.
      - If the target service requires a client certificate authentication, choose *ClientCertificateAuthentication*. See Use Destination Certificates (IDE) [page 70].
   e. Optional: In the *Properties* or *Additional Properties* section, choose the button to specify additional destination properties.

f. Save the editor.

7. When a new destination is created, the changes take effect immediately.
8. To delete a destination, choose the  button.

Related Information

Examples (IDE) [page 74]
Managing Destinations [page 54]
Encrypt Destination Passwords [page 66]

1.4.3.1.2.1.1 Encrypt Destination Passwords

Encrypt password fields for a destination configuration file.

When using a local server, the destination configuration is stored in the file system as plain text by default. The plain text storage includes password fields, which can be a security issue.

Perform the following procedure to encrypt those fields for a particular destination configuration file.

Generate a Key

To encrypt and decrypt the password fields, you need a key for an AES-128-CBC algorithm (Advanced Encryption Standard). The following steps show you how to generate this key using OpenSSL. Alternatively, you can use any other appropriate procedure.

> **i Note**
>
> If a stronger AES algorithm is required (for example, AES with 256-bit keys), you must install the JCE Unlimited Strength Jurisdiction Policy Files in the JDK/JRE.

Prerequisites

OpenSSL is provided by Linux and Mac by default. For Windows, you must install it from http://gnuwin32.sourceforge.net/packages/openssl.htm .

> **i Note**
>
> For Windows, the installer does not add the path of the *openssl.exe* file to the PATH environment variable. You should do this manually or navigate to the file before executing the OpenSSL commands in the terminal.

Procedure

1. Open a terminal.
2. (Optional) Navigate to the OpenSSL executable.
3. Adjust and execute the following command:

```
openssl enc -aes-128-cbc -P -k <pass_phrase> > <folder>\<name_of_the_key>
```

where *<pass_phrase>* is a sequence of characters, based on which the key is generated.

> ### Sample Code
>
> ```
> openssl enc -aes-128-cbc -P -k "my test pass phrase" > "E:\mykeys\Key1"
> ```

4. This procedure generates a key and stores it in the specified file (and creates the file if necessary). The key file has the following format:

```
salt=3F190F676A469E24

key=C9BA8910B87D25242AF759001842EFCF

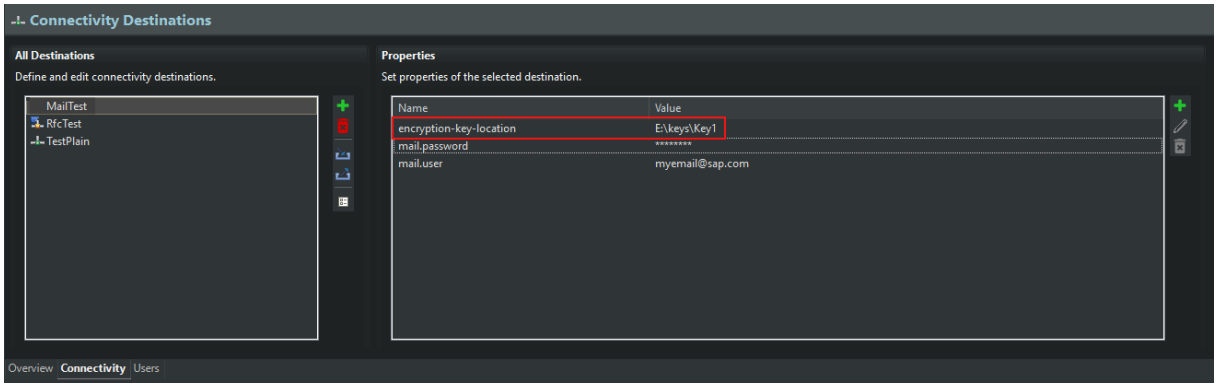
iv =AD5EE334AE9694BE96E1754B6E736C7D
```

> ### Note
>
> Only the `<key>` and `<iv>` fields are needed. If you use a different method to create the key file, you only need to include those two fields.

Configure Encryption

To store the password fields of a destination in an encrypted format, you must set the `encryption-key-location` property. The value of this property is the absolute path of the key file, containing an encryption key in the format described above.

> ### Note
>
> You should store the key file on a removable storage device. Otherwise, the decryption key can always be accessed.



When you save the destination, the destination file in the file system includes encrypted password fields. The key location, which is specified by the `encryption-key-location` property, is used when a destination is retrieved by an application to get the key and encrypt the password fields. This is done automatically by the service.

Encryption/Decryption Failure

Destination Editor (Eclipse)

- *Encryption*
  Encryption is performed when the destination is saved to the file system. If an error occurs, the `Save` operation fails and a message shows the cause.
- *Decryption*
  The following error cases may occur. If:
  - a key file is missing in the file system, the editor lets you edit the destination and specify a new location of the key.

    > **i Note**
    >
    > The `Save` operation fails until a valid key (which can decrypt the loaded destination) is provided. We strongly recommend that you provide the new location of the key immediately and save the destination. Then you can continue working with the destination as usual.

  - a key file is corrupted, the editor treats it as if the key was not found. You can specify a new location and, if the key is valid, continue working with the destination.
  - a particular field (or multiple fields) cannot be decrypted, the editor loads the destination and changes the value of the failed properties to **`blank`**. In this case, you must modify (specify new values) or remove each of these fields to fix the corrupted data.
  - the initialization of the decrypting library fails, all password fields are changed to **`blank`**.

SDK

- *Decryption*
  If decryption fails, the retrieval of an encrypted destination always causes an exception, no matter the cause of the failure. This exception is either *IllegalStateException* (if the failure is caused by a Java problem), or *IllegalArgumentException* (if the failure is caused by a problem in the destination or key file).

> **i Note**
>
> The SDK does not perform any destination encryption.

1.4.3.1.2.2 Create and Delete Destinations on the Cloud

Context

The procedure below demonstrates how you can create and configure connectivity destinations (`HTTP`, `Mail`, or `RFC`) on SAP BTP.

Procedure

1. In the context menu of the *Servers* view, choose *New* > *Server*.
2. Choose *SAP Cloud Platform* as the type of server you want to create, choose *Next*, and then *Finish*.
3. A new server `<application>.<subaccount> [Stopped]>` appears on the *Servers* view.
4. Double-click the added server to open the server editor.
5. Go to the *Connectivity* tab view.
    a. In the *All Destinations* section, choose the button to create a new destination.
    b. From the dialog window, enter a name for your destination, select its type and the choose *OK*.
    c. In the *URL* field, enter the URL of the target service to which the destination should refer.
    d. In the *Authentication* dropdown box, choose the authentication type required by the target service to authenticate the calls.
        - If the target service does not require authentication, choose *NoAuthentication*.
        - If the target service requires basic authentication, choose *BasicAuthentication*. You need to enter a user name and a password.
        - If the target service requires a client certificate authentication, choose *ClientCertificateAuthentication*. See Use Destination Certificates (IDE) [page 70].
        - If the target service requires your cloud user authentication, choose *PrincipalPropagation*. You also need to select *Proxy Type:* **`OnPremise`** and should enter the additional property *CloudConnectorVersion* with value **`2`**.
    e. In the *Proxy Type* dropdown box, choose the required type of proxy connection.

        > **i Note**
        >
        > This dropdown box allows you to choose the type of your proxy and is only available when deploying on SAP BTP. The default value is *Internet*. In this case, the destination uses the HTTP proxy for the outbound communication with the Internet. For consumption of an on-premise target service, choose the *OnPremise* option so that the proxy to the SSL tunnel is chosen and the tunnel is established to the connected Cloud Connector.

    f. Optional: In the *Properties* or *Additional Properties* section, choose the button to specify additional destination properties.
    g. Save the editor. This saves the specified destination configuration in SAP BTP.

6. When new destinations are created, the changes take effect immediately.

> **i Note**
>
> Bear in mind that changes are currently cached with a cache expiration of up to 4 minutes. That's why if you modify a destination configuration, the changes might not take effect immediately. However, if the relevant Web application is restarted on the cloud, the destination changes will take effect immediately.

7. To delete a destination, choose the  button.

Related Information

Examples (IDE) [page 74]
Managing Destinations [page 54]

1.4.3.1.2.3 Use Destination Certificates (IDE)

Prerequisites

You have opened the *Connectivity* editor in the Eclipse IDE.

Context

You can maintain keystore certificates in the *Connectivity* editor. You can upload, add and delete certificates for your connectivity destinations. Bear in mind that:

- You can use JKS, PFX and P12 files for destination keystore, and JKS, CRT, CER, DER files for destination truststore.
- You add certificates in a keystore file and then you upload, add, or delete this keystore.
- You can add certificates only for **HTTPS** destinations. Keystore is available only for `ClientCertificateAuthentication`.

Procedure

Uploading Certificates

1. Press the *Upload/Delete keystore* button. You can find it in the *All Destinations* section in the *Conectivity* editor.
2. Choose *Upload Keystore* and select the certificate you want to upload. Choose *Open* or double-click the certificate.

The certificate file is added.

> **i** Note
>
> You can upload a certificate during creation or editing of a destination, by choosing *Manage Keystore* or by pressing the *Upload/Delete keystore* button.

Adding Certificates to Destinations

1. Create a new destination, or open an existing one for editing.
2. In the *URL* field, enter an HTTPS address.
3. You can use the default JDK truststore or select another one from the truststore dropdown menu. If the menu is empty, you can upload a certificate on the fly. The password is used for the keystore that will contain your certificate on the cloud.
4. In the *Authentication* field, select *ClientCertificateAuthentication*.
5. In the *Keystore name* field, select the certificate you just added. Enter the password.

Deleting Certificates

1. Press the *Upload/Delete keystore* button.
2. Select the certificate you want to remove and choose *Delete Selected*
3. Upload another certificate, or close the *Manage Keystores* window.

Related Information

Create and Delete Destinations Locally [page 65]
Create and Delete Destinations on the Cloud [page 69]
Import Destinations (IDE) [page 72]

1.4.3.1.2.4 Import Destinations (IDE)

Prerequisites

You have previously created a connectivity destination (`HTTP`, `Mail`, or `RFC`).

> **i** Note
>
> The *Connectivity* editor allows importing destination files with extension **`.props`**, **`.properties`**, and **`.txt`**, as well as files with no extension. Destination files must be encoded in **ISO 8859-1** character encoding.

Procedure

1. On the *Servers* view, double-click your server to open its editor.
2. Go to the *Connectivity* tab view.
3. Choose button  (*Import destination*).
4. Browse to one of the following file types and choose *OK*.
   - If the destination does not contain client certificate authentication, it is saved as a single configuration file.
   - If the destination provides client certificate data, it is saved as an archive, which contains the main configuration file and a Keystore file.
5. The destination file is imported within the *Connectivity* editor.

   > **i** Note
   >
   > If the properties file contains incorrect properties or values, for example wrong destination type, the editor only displays the valid ones in the *Properties* table.

Related Information

Examples (IDE) [page 74]

1.4.3.1.2.5 Export Destinations (IDE)

Prerequisites

You have imported or created a new destination (`HTTP`, `Mail`, or `RFC`) in the Eclipse IDE.

Procedure

1. On the *Servers* view, double-click your server to open its editor.
2. Go to the *Connectivity* tab view.
3. From the list of destination names, select the one you want to export.
4. Choose button  (*Export destination*).
5. Browse to the directory you want to export your destination.
   - If the destination does not contain client certificate authentication, it is saved as a single configuration file.
   - If the destination provides client certificate data, it is saved as an archive, which contains the main configuration file and a Keystore file.

> → Tip
>
> You can keep the default name of the destination, or rename it to avoid overriding with previous files with the same name.

Next Steps

After exporting the destination, you can open it to check its content. Bear in mind that all password fields will be commented (with # symbols), and their values - deleted.

Example:

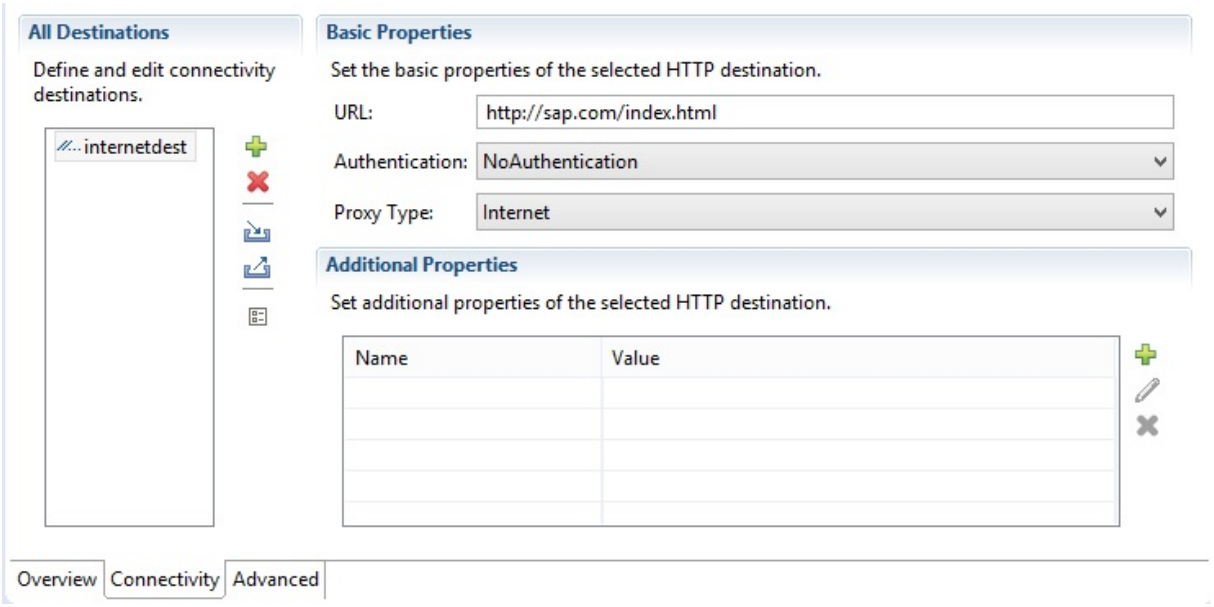
```
#Exported connectivity destination
#The following fields with passwords were removed:
#Password
#Tue Apr 21 15:01:02 FET 2015
Type=HTTP
Authentication=BasicAuthentication
Name=mydestination
URL=https://sap.com/index.html
User=p1234567890
```

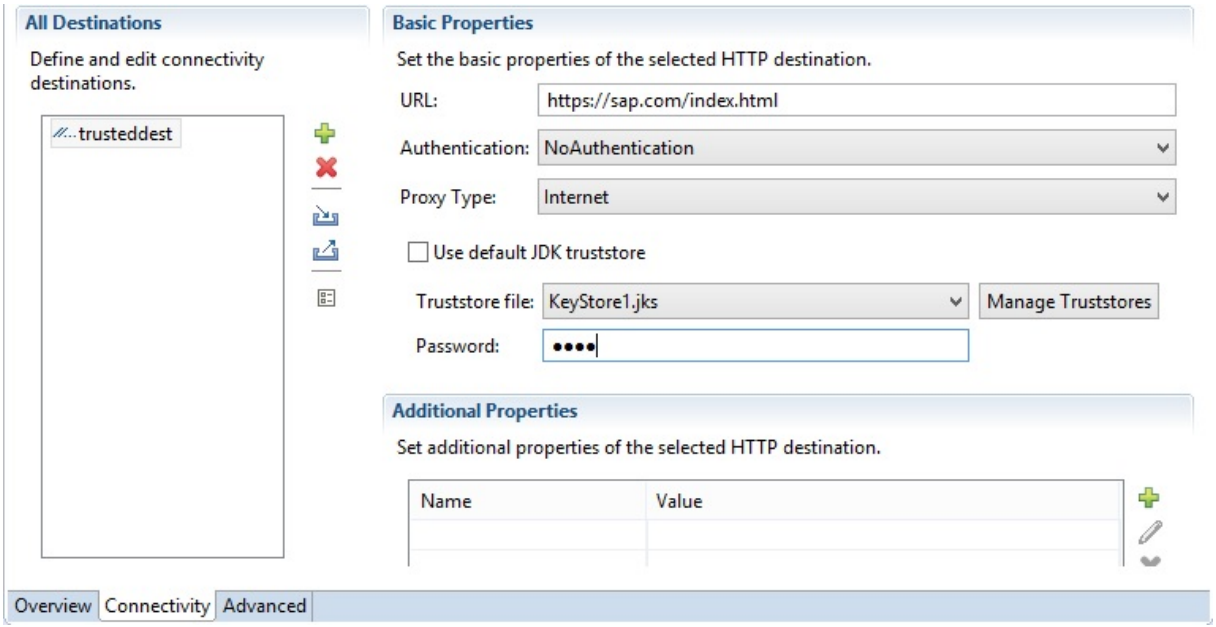
Related Information

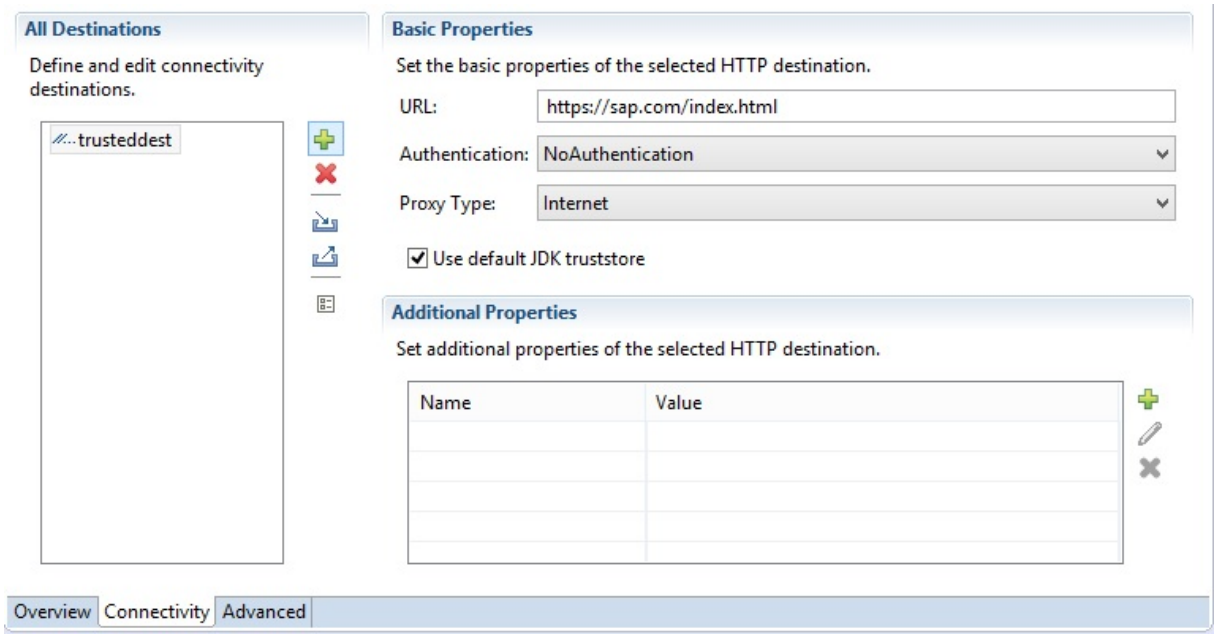
Examples (IDE) [page 74]

1.4.3.1.2.6 Examples (IDE)

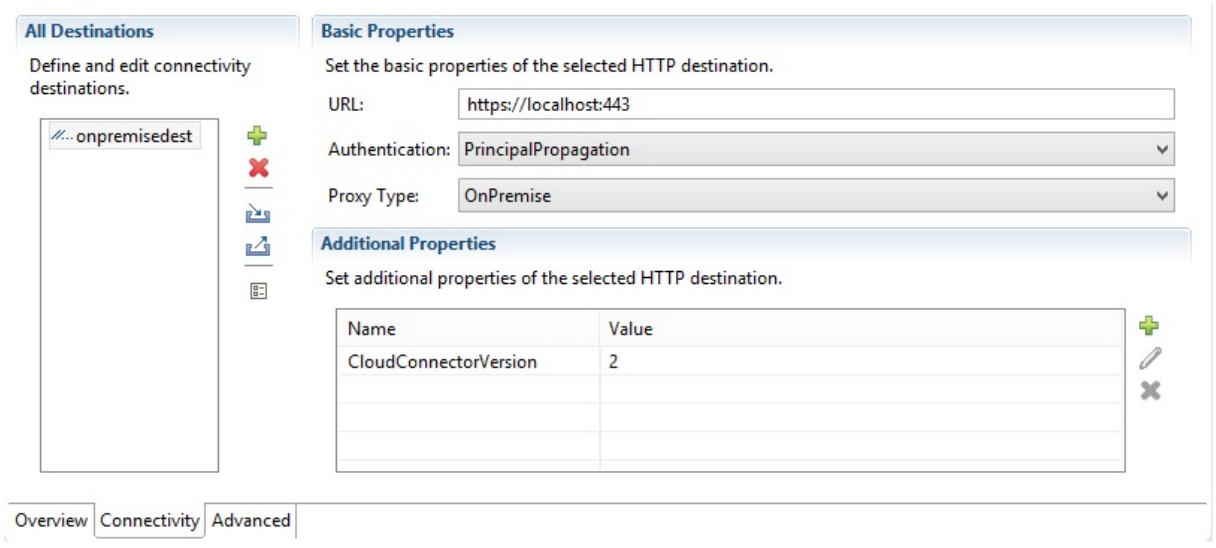
Example of HTTP Destination (Internet)



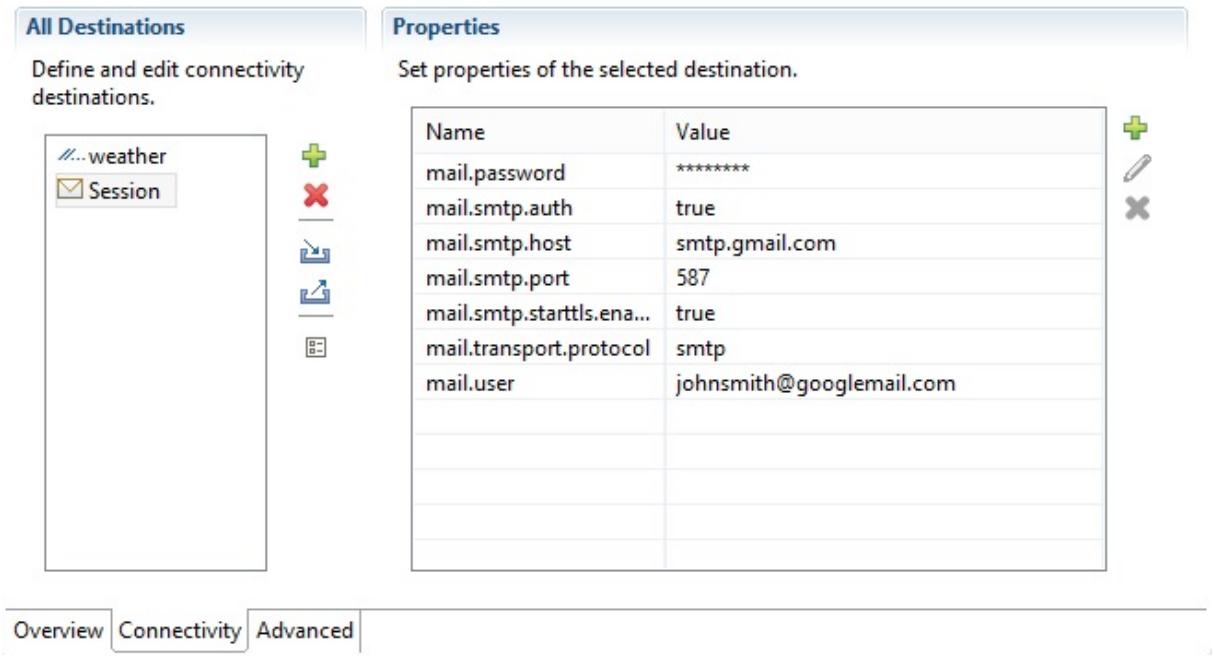




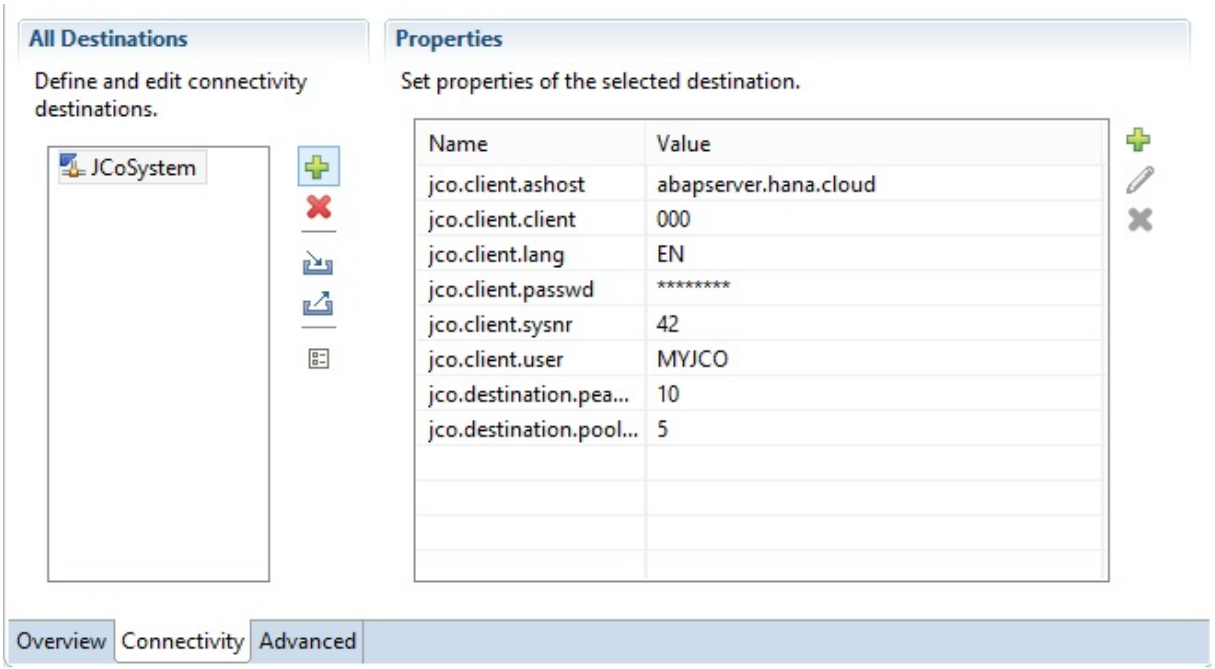
Example of HTTP Destination (OnPremise)



Example of Mail Destination



Example of RFC Destination



1.4.3.1.3 Configure Destinations from the Cockpit

Use the *Destinations* editor in SAP BTP cockpit to configure `HTTP`, `Mail`, `RFC`, and `LDAP` destinations in order to:

- Connect your cloud application to the Internet or make it consume an on-premise back-end system via HTTP(S).
- Send an e-mail from a simple Web application using an e-mail provider that is accessible on the Internet.
- Make your cloud application invoke a function module in an on-premise ABAP system via RFC.
- Use LDAP-based user authentication for your cloud application.

You can create, delete, clone, modify, import and export destinations.

Use this editor to work with destinations on subscription, subaccount, and application level.

> **i** Note
>
> Destination files must be encoded in **ISO 8859-1** character encoding.

Prerequisites

1. You have logged into the cockpit from the SAP BTP landing page, depending on your subaccount type. For more information, see Regions and Hosts Available for the Neo Environment [page 17].
2. Depending on the level you need to make destination configurations from the *Destinations* editor, make sure the following is fulfilled:
   - Subscription level – you need to have at least one application subscribed to your subaccount.
   - Application level – you need to have at least one application deployed on your subaccount.
   - Subaccount level – no prerequisites.

   For more information, see Access the Destinations Editor (Neo Environment) [page 78].

Tasks

- Create Destinations (Cockpit) [page 79]
- Check the Availability of a Destination (Cockpit) [page 83]
- Import Destinations (Cockpit) [page 88]
- Clone Destinations (Cockpit) [page 85]
- Export Destinations (Cockpit) [page 88]
- Edit and Delete Destinations (Cockpit) [page 85]

Scenarios

- Example: Send E-Mails [page 211]
- Consume Internet Services (Java Web or Java EE 6 Web Profile) [page 155]
- Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170]
- Invoke ABAP Function Modules in On-Premise ABAP Systems [page 194]

Related Information

Examples (Cockpit) [page 89]

1.4.3.1.3.1 Access the Destinations Editor (Neo Environment)

Prerequisites

- You have logged into the cockpit from the SAP BTP landing page, depending on your global account type. For more information, see Regions and Hosts Available for the Neo Environment [page 17].

Procedure

Access on Subscription Level

1. In the cockpit, select your subaccount name from the *Subaccount* menu in the breadcrumbs.
2. From the left-side navigation, choose *Applications* > *Subscriptions* to open the page with your currently subscribed Java applications (if any).
3. Select the application for which you need to create a destination.
4. From the left-side panel, choose *Destinations*.

Access on Subaccount Level

1. In the cockpit, select your subaccount name from the *Subaccount* menu in the breadcrumbs.
2. From the left-side navigation, choose *Connectivity* > *Destinations*.
3. The *Destinations* editor is opened.

Access on Application Level

1. In the cockpit, select your subaccount name from the *Subaccount* menu in the breadcrumbs.
2. From the left-side navigation, choose *Applications* > *Java Applications* to open the page with your currently deployed Java Web applications (if any).
3. Select the application for which you need to create a destination.
4. From the left-side panel, choose *Configuration* > *Destinations*.
5. The *Destinations* editor is opened.

Related Information

Create Destinations (Cockpit) [page 79]
Import Destinations (Cockpit) [page 88]
Edit and Delete Destinations (Cockpit) [page 85]

1.4.3.1.3.2 Create Destinations (Cockpit)

Prerequisites

You have logged into the cockpit and opened the *Destinations* editor.

Context

To learn how to create `HTTP`, `RFC`, and `Mail` destinations, follow the steps on the relevant pages:

- Create HTTP Destinations [page 80]
- Create RFC Destinations [page 81]
- Create Mail Destinations [page 82]

Related Information

Managing Destinations [page 54]
Examples (Cockpit) [page 89]

1.4.3.1.3.2.1 Create HTTP Destinations

Create HTTP destinations in the *Destinations* editor (SAP BTP cockpit).

Prerequisites

You have logged into the cockpit and opened the *Destinations* editor.

Procedure

1. Choose *New Destination*.
2. Enter a destination name.
3. From the *Type* dropdown menu, choose **`HTTP`**.
4. The *Description* field is optional.
5. Specify the destination URL.
6. From the *Proxy Type* dropdown box, select **`Internet`** or **`OnPremise`**, depending on the connection you need to provide for your application.

   > **i Note**
   >
   > For more information, see also: HTTP Destinations [page 91].

7. From the *Authentication* dropdown box, select the authentication type you need for the connection.

   For details, see HTTP Destinations [page 91].

   > **i Note**
   >
   > If you set an **`HTTPS`** destination, you need to also add a Trust Store. For more information, see Use Destination Certificates (Cockpit) [page 86].

8. (Optional) If you are using more than one Cloud Connector for your subaccount, you must enter the *Location ID* of the target Cloud Connector.
   See also Managing Subaccounts [page 289] (section **Procedure**, step 4).
9. (Optional) You can enter additional properties.
   a. In the *Additional Properties* panel, choose *New Property*.
   b. Enter a key (name) or choose one from the dropdown menu and specify a value for the property. You can add as many properties as you need.
   c. To delete a property, choose the ✕ button next to it.

   > **i Note**
   >
   > For a detailed description of specific properties for SAP Business Application Studio (formerly known as SAP Web IDE), see Connecting to External Systems.

10. When you are ready, choose the *Save* button.

Related Information

Examples (Cockpit) [page 89]
HTTP Destinations [page 91]
Edit and Delete Destinations (Cockpit) [page 85]

1.4.3.1.3.2.2 Create RFC Destinations

Create RFC destinations in the *Destinations* editor (SAP BTP cockpit).

Prerequisites

You have logged into the cockpit and opened the *Destinations* editor.

Procedure

1. Choose *New Destination*.
2. Enter a destination name.
3. From the *Type* dropdown menu, choose **`RFC`**.
4. The *Description* field is optional.
5. From the *Proxy Type* dropdown box, select **`Internet`** or **`OnPremise`**, depending on the connection you need to provide for your application.

   > **i Note**
   >
   > Using `<Proxy Type>` **`Internet`** , you can connect your application to any target service that is exposed to the Internet. `<Proxy Type>` **`OnPremise`** requires the Cloud Connector to access resources within your on-premise network.

6. Enter credentials for *User* and *Password*.
7. (Optional) Enter an *Alias User* name. See also User Logon Properties [page 113].
8. (Optional) Enter credentials for *Repository User* and *Repository Password*, if you want to use a different user for repository lookups. See also Repository Configuration [page 117].
9. (Optional) If you are using more than one Cloud Connector for your subaccount, you must enter the *Location ID* of the target Cloud Connector.
   See also Managing Subaccounts [page 289] (section **Procedure**, step 4).

10. (Optional) You can enter additional properties.
    a. In the *Additional Properties* panel, choose *New Property*.
    b. Enter a key (name) or choose one from the dropdown menu and specify a value for the property. You can add as many properties as you need.
    c. To delete a property, choose the  button next to it.

    If you add `PrincipalPropagation` as additional property (`jco.destination.auth_type`), your RFC destination must not contain user and password information. It can, however, contain repository credentials.

    > **i Note**
    >
    > For a detailed description of RFC-specific properties (JCo properties), see RFC Destinations [page 113].

11. When you are ready, choose the *Save* button.

Related Information

Examples (Cockpit) [page 89]
RFC Destinations [page 113]
Edit and Delete Destinations (Cockpit) [page 85]
Cloud Connector (Neo Environment) [page 232]

1.4.3.1.3.2.3 Create Mail Destinations

Prerequisites

You have logged into the cockpit and opened the *Destinations* editor.

Procedure

1. Choose *New Destination*.
2. Enter a destination name.
3. From the *Type* dropdown menu, choose **`MAIL`**.
4. The *Description* field is optional.
5. Enter credentials for *User* and *Password*.
6. Optional: You can enter additional properties.

a. In the *Additional Properties* panel, choose *New Property*.
b. Enter a key (name) or choose one from the dropdown menu and specify a value for the property. You can add as many properties as you need.
c. To delete a property, choose the  button next to it.

7. When you are ready, choose the *Save* button.

Related Information

Examples (Cockpit) [page 89]
Mail Destinations [page 127]
Edit and Delete Destinations (Cockpit) [page 85]

1.4.3.1.3.3 Check the Availability of a Destination (Cockpit)

Prerequisites

You have logged into the cockpit and opened the *Destinations* editor.

Context

You can use the *Check Connection* button in the *Destinations* editor of the cockpit to verify if the *URL* configured for an HTTP Destination is reachable and if the connection to the specified system is possible.

> **i Note**
>
> This check is available with *Cloud Connector version 2.7.1 or higher*.

For each destination, the check button is available in the destination detail view and in the destination overview list (icon *Check availability of destination connection* in section *Actions*).

> **i Note**
>
> The check does not guarantee that a backend is operational. It only verifies if a connection to the backend is possible.

This check is supported only for destinations with *Proxy Type* `Internet` and `OnPremise`:

- For *Internet* destinations:

- If the check receives an **HTTP status code above or equal to 500** from the targeted URL, the check is considered failed.
- Every HTTP status code below 500 is treated as successful.

- For *OnPremise* destinations:
  - If the targeted backend is reached and returns an **HTTP status code below 500** the check is considered successful.

Error Messages for OnPremise Destinations

Error Message	Reason	Action
Backend status could not be determined.	• The Cloud Connector version is less than 2.7.1. • The Cloud Connector is not connected to the subaccount. • The backend returns a HTTP status code above or equal to 500 (server error). • The Cloud Connector is not configured properly.	• Upgrade the Cloud Connector to version 2.7.1 or higher. • Connect the Cloud Connector to the corresponding subaccount. • Check the server status (availability) of the back-end system. • Check the basic Cloud Connector configuration steps: Initial Configuration [page 278]
Backend is not available in the list of defined system mappings in Cloud Connector.	The Cloud Connector is not configured properly.	Check the basic Cloud Connector configuration steps: Initial Configuration [page 278]
Resource is not accessible in Cloud Connector or backend is not reachable.	The Cloud Connector is not configured properly.	Check the basic Cloud Connector configuration steps: Initial Configuration [page 278]
Backend is not reachable from Cloud Connector.	Cloud connector configuration is ok but the backend is not reachable.	Check the backend (server) availability.

1.4.3.1.3.4 Clone Destinations (Cockpit)

Prerequisites

You have previously created or imported a connectivity destination (`HTTP`, `Mail`, or `RFC`) in the *Destinations* editor of the cockpit.

Procedure

1. In the *Destinations* editor, go to the existing destination which you want to clone.
2. Choose the  icon.
3. The editor automatically creates and opens a new destination that contains all the properties of the selected one.
4. You can modify some parameters if you need.
5. When you are ready, choose the *Save* button.

Related Information

Examples (Cockpit) [page 89]
Export Destinations (Cockpit) [page 88]

1.4.3.1.3.5 Edit and Delete Destinations (Cockpit)

Prerequisites

You have previously created or imported a connectivity destination (`HTTP`, `Mail`, or `RFC`) in the *Destinations* editor of the cockpit.

Procedure

- Edit a destination:

  1. To edit a created or imported destination, choose the  button.
  2. You can edit the main parameters as well as the additional properties of a destination.
  3. Choose the *Save* button. The changes will take effect in up to five minutes.

  > → Tip
  >
  > For complete consistency, we recommend that you first stop your application, then apply your destination changes, and then start again the application. Also, bear in mind that these steps will cause application downtime.

- Delete a destination:

  To remove an existing destination, choose the  button. The changes will take effect in up to five minutes.

Related Information

Examples (Cockpit) [page 89]
Export Destinations (Cockpit) [page 88]

1.4.3.1.3.6 Use Destination Certificates (Cockpit)

Prerequisites

You have logged into the cockpit and opened the *Destinations* editor. For more information, see Access the Destinations Editor (Neo Environment) [page 78].

Context

This page explains how you can maintain truststore and keystore certificates in the *Destinations* editor. You can upload, add and delete certificates for your connectivity destinations. Bear in mind that:

- You can only use JKS, PFX and P12 files for destination key store, and JKS, CRT, CER, DER for destination trust store.
- You can add certificates only for **HTTPS** destinations. Truststore can be used for all authentication types. Keystore is available only for `ClientCertificateAuthentication`.
- An uploaded certificate file should contain the entire certificate chain.

Procedure

Uploading Certificates

1. Choose the *Certificates* button.
2. Choose *Upload Certificate*.
3. Browse to the certificate file you need to upload.
   The certificate file is added.

> **i** Note
>
> You can upload a certificate during creation or editing of a destination, by clicking the *Upload and Delete Certificates* link.

Adding Certificates to Destinations

1. Create a new destination, or open an existing one for editing.
2. In the *URL* field, enter an HTTPS address.
3. You can use the default JDK truststore or select another one from the dropdown menu. If the menu is empty, you can upload a certificate on the fly. To omit this property, you can set `TrustAll=true`.
4. If you choose *Authentication* = **`ClientCertificateAuthentication`**, you need to also provide a keystore.

Deleting Certificates

1. Choose the *Certificates* button or click the *Upload and Delete Certificates* link.
2. Select the certificate you want to remove and choose *Delete Selected*.
3. Upload another certificate, or close the *Certificates* window.

Related Information

Create Destinations (Cockpit) [page 79]
Import Destinations (Cockpit) [page 88]
Edit and Delete Destinations (Cockpit) [page 85]

1.4.3.1.3.7 Import Destinations (Cockpit)

Prerequisites

You have previously created a connectivity destination (`HTTP`, `Mail`, or `RFC`).

> **i** Note
>
> The *Destinations* editor allows importing destination files with extension **`.props`**, **`.properties`**, **`.jks`**, and **`.txt`**, as well as files with no extension. Destination files must be encoded in **ISO 8859-1** character encoding.

Procedure

1. Log into the cockpit and open the *Destinations* editor.
2. Choose *Import from File*.
3. Browse to a configuration file that contains destination configuration.
   - If the configuration file contains valid data, it is displayed in the *Destinations* editor with no errors. The *Save* button is enabled so that you can successfully save the imported destination.
   - If the configuration file contains invalid properties or values, under the relevant fields in the *Destinations* editor are displayed error messages in red which prompt you to correct them accordingly.

Related Information

Examples (Cockpit) [page 89]
Edit and Delete Destinations (Cockpit) [page 85]

1.4.3.1.3.8 Export Destinations (Cockpit)

Export destinations from the *Destinations* editor in the SAP BTP cockpit to backup or reuse a destination configuration.

Prerequisites

You have created a connectivity destination (`HTTP`, `Mail`, or `RFC`) in the *Destinations* editor.

Procedure

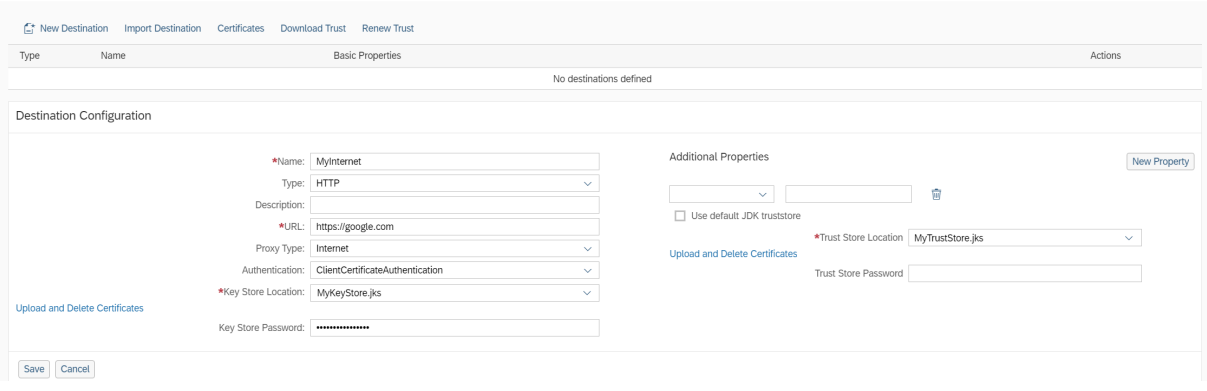
1. Log into the cockpit and open the *Destinations* editor.
2. Select a destination and choose the button.
3. Browse to the location on your local file system where you want to save the new destination.
    - If the destination does not contain client certificate authentication, it is saved as a single configuration file.
    - If the destination provides client certificate data, it is saved as an archive, which contains the main configuration file and a JKS file.

Related Information

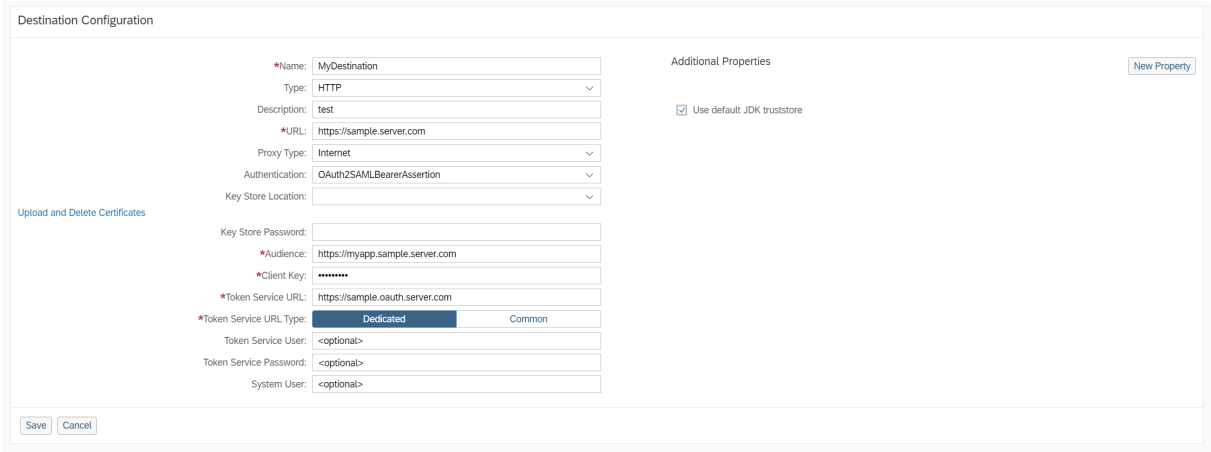
Examples (Cockpit) [page 89]
Edit and Delete Destinations (Cockpit) [page 85]

1.4.3.1.3.9 Examples (Cockpit)

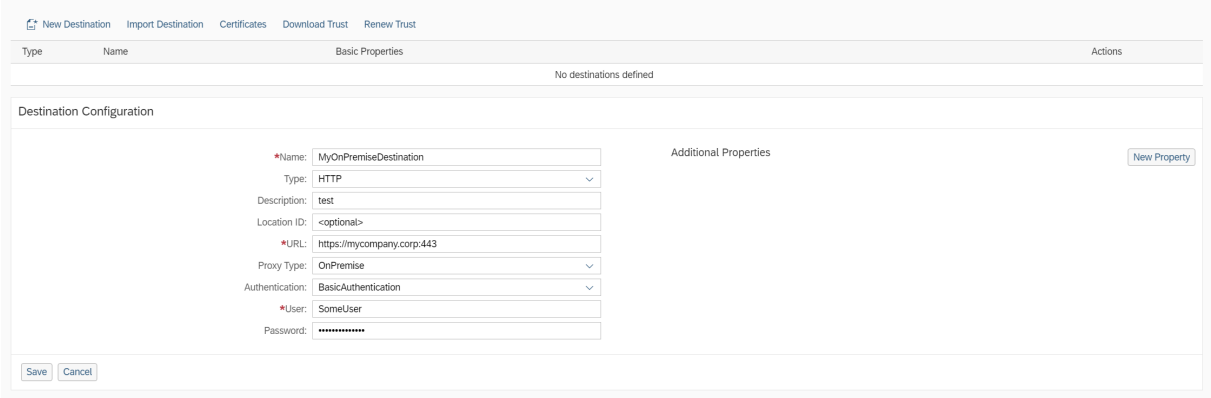
Example of HTTP Destination (Internet, Client Certificate Authentication)



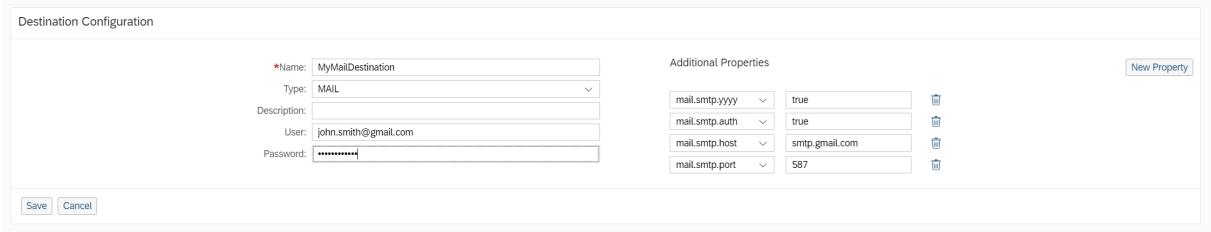
Example of HTTP Destination (Internet, OAuth2SAMLBearerAssertion)



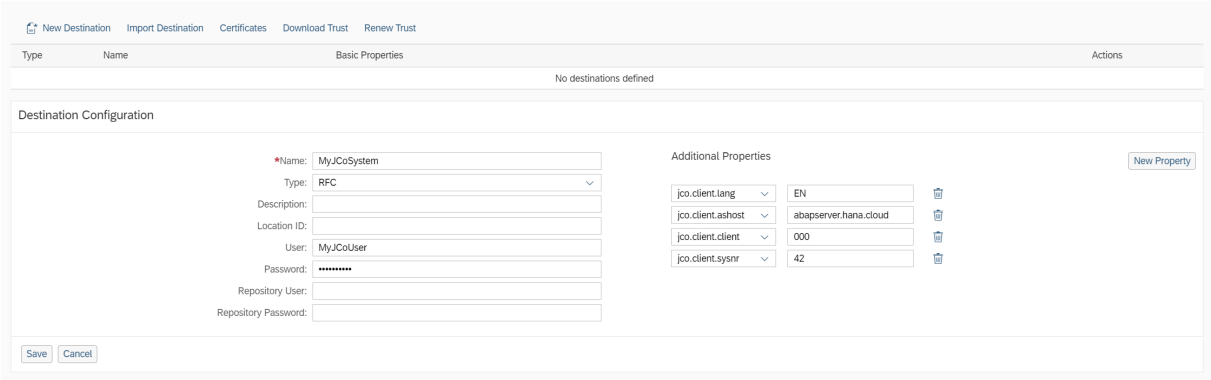
Example of HTTP Destination (On-Premise)



Example of Mail Destination



Example of RFC Destination



The following main properties correspond to the relevant additional properties:

User → `jco.client.user`

Password → `jco.client.passwd`

Repository password → `jco.destination.repository.passwd`

> **i** Note
>
> For security reasons, do not use these additional properties but use the corresponding main properties' fields.

Related Information

HTTP Destinations [page 91]
RFC Destinations [page 113]
Mail Destinations [page 127]

1.4.3.2 HTTP Destinations

Overview

The HTTP destinations provide data communication via HTTP protocol and are used for both Internet and on-premise connections.

HTTP Destination Properties

The runtime tries to resolve a destination in the order: *Subscription Level* → *Subaccount Level* → *Application Level*. By using the optional "`DestinationProvider`" property, a destination can be limited to application level only, that is, the runtime tries to resolve the destination on application level.

Property	Description
`DestinationProvider`	Restricts destination to application level. If the property is specified, the destination will be searched on the application level only. By default, destinations are searched on all configuration levels.
`URL.connectionTimeoutInSeconds`	Period of time in which the HTTP client expects a confirmation after it initiated a new TCP connection. Introduced in version 2.15.
`URL.socketReadTimeoutInSeconds`	Period of time the HTTP client will wait for receiving a response (data) from the server. Introduced in version 2.15.

> **i Note**
>
> If you use Java Web Tomcat 7 runtime container, the `DestinationProvider` property is not supported. Instead, you can use AuthenticationHeaderProvider API [page 141].

Example

```
Name=weather
Type=HTTP
Authentication=NoAuthentication
DestinationProvider=Application
```

Supported Proxy Types for Connectivity

The proxy types supported by the Connectivity service are:

- *Internet* - The application can connect to an external REST or SOAP service on the Internet.
- *OnPremise* - The application can connect to an on-premise back-end system through the Cloud Connector.

The proxy type used for a destination must be specified by the destination property `ProxyType`. The property's default value (if not configured explicitly) is **`Internet`**.

Setting Proxy Types for SAP HANA Cloud Local Runtime

If you work in your local development environment behind a proxy server and want to use a service from the Internet, you need to configure your proxy settings on JVM level. To do this, proceed as follows:

1. On the *Servers* view, double-click the added server and choose *Overview* to open the editor.
2. Click the *Open Launch Configuration* link.
3. Choose the *(x)=Arguments* tab page.
4. In the *VM Arguments* box, add the following row:

```
-Dhttp.proxyHost=yourproxyHost -Dhttp.proxyPort=yourProxyPort -
Dhttps.proxyHost=yourproxyHost -Dhttps.proxyPort=yourProxyPort
```

5. Choose *OK*.
6. Start or restart your SAP HANA Cloud local runtime.

For more information and example, see Consume Internet Services (Java Web or Java EE 6 Web Profile) [page 155].

Setting Proxy Types for SAP HANA Cloud

- When using the *Internet* proxy type, you do not need to perform any additional configuration steps.
- When using the *OnPremise* proxy type, you configure the setting the standard way through the *Connectivity* editor in the Eclipse IDE.
  For more information and example, see Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170].

Configuring Authentication

When creating an HTTP destination, you can use different authentication types for access control::

- Server Certificate Authentication [page 93]
- SAP Assertion SSO Authentication [page 96]
- Principal Propagation SSO Authentication for HTTP [page 99]
- SAML Bearer Assertion Authentication [page 101]
- Application-to-Application SSO Authentication [page 104]
- Client Authentication Types for HTTP Destinations [page 108]
- OAuth Client Credentials Authentication [page 110]

1.4.3.2.1 Server Certificate Authentication

Context

The server certificate authentication is applicable for all client authentication types, described below.

> **i Note**
>
> TLS 1.2 became the default TLS version of HTTP destinations. If an HTTP destination is consumed by a java application the change will be effective after restart. All HTTP destinations that use the HTTPS protocol and have ProxyType=Internet can be affected. Previous TLS version can be used by configuring an additional property TLSVersion=TLSv1.0 or TLSVersion=TLSv1.1.

Properties

Property	Description
`TLSVersion`	Optional property. Can be used to specify the preferred TLS version to be used by the current destination. Since TLS 1.2 is not enabled by default on the older java versions this property can be used to configure TLS 1.2 in case this is required by the server configured in this destination. It is usable only in HTTP destinations. Example: `TLSVersion=`**`TLSv1.2`** .
`TrustStoreLocation` 1. When used in local environment 2. When used in cloud environment	Path to the JKS file which contains trusted certificates (Certificate Authorities) for authentication against a remote client. 1. The relative path to the JKS file. The root path is the server's location on the file system. 2. The name of the JKS file. **i Note** The default JDK trust store is appended to the trust store defined in the destination configuration. As a result, the destination simultaneously uses both trust stores. If the `TrustStoreLocation` property is not specified, the JDK trust store is used as a default trust store for the destination.
`TrustStorePassword`	Password for the JKS trust store file. This property is mandatory if `TrustStoreLocation` is used.
`TrustAll`	If this property is set to **`TRUE`** in the destination, the server certificate will not be checked for SSL connections. It is intended for test scenarios only, and should not be used in production (since the SSL server certificate is not checked, the server is not authenticated). The possible values are **`TRUE`** or **`FALSE`**; the default value is **`FALSE`** (that is, if the property is not present at all). In case `TrustAll` = **`TRUE`**, the `TrustStoreLocation` property is ignored so you can omit it. In case `<TrustAll>` = **`FALSE`**, the `<TrustStoreLocation>` property is mandatory to be used.

Property	Description
`HostnameVerifier`	Optional property. It has two values: **`Strict`** and **`BrowserCompatible`**. This property specifies how the server hostname matches the names stored inside the server's X.509 certificate. This verifying process is only applied if TLS or SSL protocols are used and is not applied if the `TrustAll` property is specified. The default value (used if no value is explicitly specified) is **`Strict`**. • `Strict HostnameVerifier` works in the same way as Oracle Java 1.4, Oracle Java 5, and Oracle Java 6-rc. It is also similar to Microsoft Internet Explorer 6. This implementation appears to be compliant with RFC 2818 for dealing with wildcards. A wildcard such as "*.foo.com" matches only subdomains at the same level, for example "a.foo.com". It does not match deeper subdomains such as "a.b.foo.com". • `BrowserCompatible HostnameVerifier` works in the same way as Curl and Mozilla Firefox. The hostname must match either the first common name (CN) or any of the subject-alts. A wildcard can occur in the CN and in any of the subject-alts. The only difference between **`BrowserCompatible`** and **`Strict`** is that a wildcard (such as ".foo.com") with **`BrowserCompatible`** matches all subdomains, including "a.b.foo.com". For more information about these Java classes, see Package org.apache.http.conn.ssl . In case `<TrustAll>` = **`TRUE`**, the `<HostnameVerifier>` property is ignored so you can omit it.

> **Note**
>
> You can upload trust store JKS files using the same command as for uploading destination configuration property files. You only need to specify the JKS file instead of the destination configuration file.

> **Note**
>
> Connections to remote services which require *Java Cryptography Extension (JCE) unlimited strength jurisdiction policy* are not supported.

Configuration

- Configure Destinations from the Cockpit [page 77]
- Configure Destinations from the Eclipse IDE [page 64]
- Configure Destinations from the Console Client [page 56]

Related Information

Client Authentication Types for HTTP Destinations [page 108]

1.4.3.2.2 SAP Assertion SSO Authentication

Create and configure an SAP Assertion SSO destination for an application in the Neo environment.

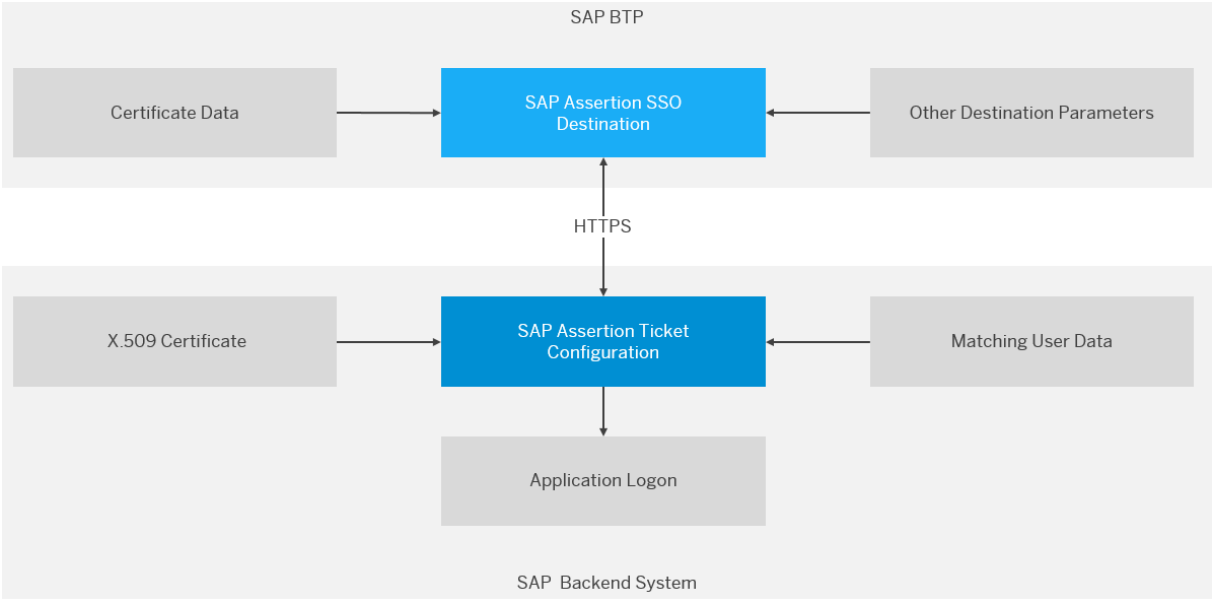
> ⚠ Caution
>
> Authentication type SAP Assertion SSO is deprecated. Use Principal Propagation SSO Authentication for HTTP [page 99] instead, which is the recommended mechanism for establishing single sign-on (SSO).

Context

By default, all SAP systems accept SAP assertion tickets for user propagation.

> ℹ Note
>
> For more information, see Authentication Assertion Tickets.

The aim of the *SAPAssertionSSO* destination is to generate such an assertion ticket in order to propagate the currently logged-on SAP BTP user to an SAP backend system. You can only use this authentication type if the user IDs on both sides are the same. The following diagram shows the elements of the configuration process on the SAP BTP and in the corresponding backend system:



Configuration Steps

1. Configure the back-end system so that it can accept SAP assertion tickets signed by a trusted `x.509` key pair. For more information, see Configuring a Trust Relationship for SAP Assertion Tickets.
2. Create and configure a *SAPAssertionSSO* destination by using the properties listed below, and deploy it on SAP BTP.
   - Configure Destinations from the Cockpit [page 77]
   - Configure Destinations from the Console Client [page 56]

> **i Note**
>
> Configuring *SAPAssertionSSO* destinations from the Eclipse IDE is not yet supported.

Properties

The following credentials need to be specified:

Property	Description
`Name`	Destination name. It must be the same as the destination name you use for the configuration tools, that is, the console client and *Destinations* editor (cockpit).
`Type`	Destination type. Use **`HTTP`** for all HTTP(S) destination.
`URL`	URL of the protected resource on the called application
`Authentication`	Authentication type. Use **`SAPAssertionSSO`** as a value.
`IssuerSID`	This system ID should be trusted by the back-end system.
`IssuerClient`	This client ID should be trusted by the back-end system.
`RecipientSID`	System ID (SID) of the back-end system
`RecipientClient`	Client ID of the back-end system
`Certificate`	A base64 encoded certificate that is trusted by the SAP system
`SigningKey`	A base64 encoded signing/private key that is trusted by the SAP system
`SystemUser`	Optional property. • If specified, all SAP assertion tickets are generated with the specified user ID. • If not specified, all SAP assertion tickets are sent on behalf of the currently logged-on user. Thus, if the current user needs to be propagated, do not use this property.
`ProxyType`	You can use both proxy types **`Internet`** and **`OnPremise`**.

Property	Description
`CloudConnectorLocationId`	Optional property. As of Cloud Connector version 2.9.0, you can connect multiple Cloud Connectors to an account as long as their location ID is different. The value defines the location ID identifying the Cloud Connector over which the connection is opened. The default value is an empty string identifying the Cloud Connector that is connected without any location ID, which is also the case for all Cloud Connector versions prior to 2.9.0.

`URL.headers.<header-key>`	Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a `URL.headers` prefix separated by dot-delimiter. For example:
	
	Sample Code
	`URL.headers.<header-key-1>=<header-value-1>`
	`...`
	`URL.headers.<header-key-n>=<header-value-n>`
	
	Note
	This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention. The HttpDestination (Version 2) [page 149] library supports this as of version 2.14.

Example

```
Name=weather
Type=HTTP
Authentication=SAPAssertionSSO
IssuerSID=JAV
IssuerClient=000
RecipientSID=SAP
RecipientClient=100
Certificate=MIICiDCCAkegAwI...rvHTQ\=\=
SigningKey=MIIBSwIB...RuqNKGA\=
```

1.4.3.2.3 Principal Propagation SSO Authentication for HTTP

Forward the identity of a cloud user from a Neo application to a backend system to enable single sign-on (SSO).

Context

A *PrincipalPropagation* destination enables single sign-on (SSO) by forwarding the identity of a cloud user to the Cloud Connector, and from there to the target on-premise system. In this way, the cloud user's identity can be provided without manual logon.

> **i** Note
>
> This authentication type applies only for on-premise connectivity.

Configuration Steps

You can create and configure a *PrincipalPropagation* destination by using the properties listed below, and deploy it on SAP BTP. For more information, see:

- Configure Destinations from the Cockpit [page 77]
- Configure Destinations from the Eclipse IDE [page 64]
- Configure Destinations from the Console Client [page 56]

Properties

The following credentials need to be specified:

Property	Description
`Name`	Destination name. It must be the same as the destination name you use in the configuration tools, that is, *Connectivity* editor (Eclipse IDE), *Destinations* editor (cockpit), and the console client.
`Type`	Destination type. Use **`HTTP`** for all HTTP(S) destinations.
`URL`	Virtual URL of the protected on-premise application.
`Authentication`	Authentication type. Use **`PrincipalPropagation`** as value.
`ProxyType`	You can only use proxy type **`OnPremise`**.

Property	Description
`CloudConnectorLocationId`	As of Cloud Connector 2.9.0, you can connect multiple Cloud Connectors to a subaccount as long as their location ID is different. The location ID specifies the Cloud Connector over which the connection is opened. The default value is an empty string identifying the Cloud Connector that is connected without any location ID. This is also valid for all Cloud Connector versions prior to 2.9.0.

`URL.headers.<header-key>`	Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a `URL.headers` prefix separated by dot-delimiter. For example:
	
	Sample Code
	`URL.headers.<header-key-1>=<header-value-1>`
	`...`
	`URL.headers.<header-key-n>=<header-value-n>`
	
	Note
	This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention. The HttpDestination (Version 2) [page 149] library supports this as of version 2.14.

Example

```
Name=OnPremiseDestination
Type=HTTP
URL= http://virtualhost:80
Authentication=PrincipalPropagation
ProxyType=OnPremise
```

Related Information

Principal Propagation [page 129]

1.4.3.2.4 SAML Bearer Assertion Authentication

Context

SAP BTP supports applications to use the SAML Bearer assertion flow for consuming OAuth-protected resources. As a result, applications do not need to deal with some of the complexities of OAuth and can reuse existing identity providers for user data. Users are authenticated by using SAML against the configured trusted identity providers. The SAML assertion is then used to request an access token from an OAuth authorization server. This access token is automatically injected in all HTTP requests to the OAuth-protected resources.

> → Tip
>
> The access tokens are auto-renovated. When a token is about to expire, a new token is created shortly before the expiration of the old one.

Configuration Steps

You can create and configure an *OAuth2SAMLBearerAssertion* destination by using the properties listed below, and deploy it on SAP BTP. For more information, see:

- Configure Destinations from the Cockpit [page 77] or
- Configure Destinations from the Console Client [page 56]

> ℹ Note
>
> Configuring *OAuth2SAMLBearerAssertion* destinations from the Eclipse IDE is not yet supported.

If you use the proxy type **`OnPremise`**, *both* OAuth server and the protected resource must be located on premise and exposed via the Cloud Connector. Make sure to set `URL` to the virtual address of the protected resource and `tokenServiceURL` to the virtual address of the OAuth server (see section *Properties* below).

> ℹ Note
>
> The combination *on-premise* OAuth server and protected resource on the *Internet* is not supported, as well as OAuth server on the *Internet* and protected resource *on premise*.

Properties

The table below lists the destination properties for *OAuth2SAMLBearerAssertion* authentication type. You can find the values for these properties in the provider-specific documentation of OAuth-protected services. Usually, only a subset of the optional properties is required by a particular service provider.

Property	Description
Required	
`Name`	Destination name. It must be the same as the destination name you use for the configuration tools, that is, the console client and *Destinations* editor (cockpit).
`Type`	Destination type. Use **`HTTP`** as a value for all HTTP(S) destinations.
`URL`	URL of the protected resource on the called application
`ProxyType`	You can use the proxy types **`Internet`** or **`OnPremise`**. If you use **`OnPremise`**, the OAuth server must be accessed through the Cloud Connector [page 232].
`Authentication`	Authentication type. Use **`OAuth2SAMLBearerAssertion`** as a value.
`audience`	Intended audience for the assertion, which will be verified by the OAuth authorization server. For more information, see SAML 2.0 Bearer Assertion Profiles for OAuth 2.0 .
`clientKey`	Key that identifies the consumer to the authorization server
`tokenServiceURL`	URL of the OAuth server
`tokenServiceUser`	User for basic authentication to OAuth server (if required)
`tokenServicePassword`	Password for `tokenServiceUser` (if required)
Additional	
(Deprecated) `SystemUser`	User to be used when requesting access token from the OAuth authorization server. If this property is not specified, the currently logged-in user will be used. **⚠ Caution** This property is deprecated and will be removed soon. We recommend that you work on behalf of specific (named) users instead of working with a technical user. As an alternative for technical user communication, we strongly recommend that you use one of these authentication types: • Basic Authentication (see Client Authentication Types for HTTP Destinations [page 108]) • Client Certificate Authentication (see Client Authentication Types for HTTP Destinations [page 108]) • OAuth Client Credentials Authentication [page 110] To extend an OAuth access token's validity, consider using an OAuth refresh token.

Property	Description
`nameQualifier`	Security domain of the user for which access token will be requested
`companyId`	Company identifier
`assertionIssuer`	Issuer of the SAML assertion
`authnContextClassRef`	Value of the `AuthnContextClassRef` tag, which is part of generated **`OAuth2SAMLBearerAssertion`** authentication. For more information, see SAML 2.0 specification .
`nameIdFormat`	Value of the `NameIdFormat` tag, which is part of generated **`OAuth2SAMLBearerAssertion`** authentication. For more information, see SAML 2.0 specification .
`userIdSource`	When this property is set, the generated SAML2 assertion uses the currently logged-in user as a value for the `NameId` tag.
`scope`	The value of the OAuth 2.0 `scope` parameter expressed as a list of space-delimited, case-sensitive strings.
`SkipSSOTokenGenerationWhenNoUser`	If this parameter is set and there is no user logged in, token generation is skipped, thus allowing anonymous access to public resources. If set, it may have any value.

`URL.headers.<header-key>`	Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a `URL.headers` prefix separated by dot-delimiter. For example:
	
	Sample Code
	`URL.headers.<header-key-1>=<header-value-1>`
	`...`
	`URL.headers.<header-key-n>=<header-value-n>`
	
	Note
	This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention. The HttpDestination (Version 2) [page 149] library supports this as of version 2.14.

> **i Note**
>
> When the OAuth authorization server is called, it accepts the trust settings of the destination. For more information, see Server Certificate Authentication [page 93].

Example

The connectivity destination below provides HTTP access to the OData API of the SuccessFactors Jam.

```
URL=https://demo.sapjam.com/OData/OData.svc
Name=sap_jam_odata
TrustAll=true
ProxyType=Internet
Type=HTTP
Authentication=OAuth2SAMLBearerAssertion
tokenServiceURL=https://demo.sapjam.com/api/v1/auth/token
clientKey=Aa1Bb2Cc3DdEe4F5GHIJ
audience=cubetree.com
nameQualifier=www.successfactors.com
apiKey=<apiKey>
```

Related Information

Create HTTP Destinations [page 80]
Examples (Cockpit) [page 89]

1.4.3.2.5 Application-to-Application SSO Authentication

Context

The *AppToAppSSO* destinations are used in scenario of application-to-application communication where the caller needs to propagate its logged-in user. Both applications are deployed on SAP BTP.

Configuration Steps

1. Configure your subaccount to allow principal propagation. For more information, see Configure the Local Service Provider [page 1721].

> **Note**
>
> This setting is done per subaccount, which means that once set to **`Enabled`** all applications within the subaccount will accept user propagation.

2. Create and configure an *AppToAppSSO* destination by using the properties listed below, and deploy it on SAP BTP. For more information, see:
   - Configure Destinations from the Cockpit [page 77]
   - Configure Destinations from the Console Client [page 56]

> **Note**
>
> Configuring *AppToAppSSO* destinations from the Eclipse IDE is not yet supported.

Properties

The following credentials need to be specified:

Property	Description
`Name`	Destination name. It must be the same as the destination name you use for the configuration tools, that is, the console client and *Destinations* editor (cockpit).
`Type`	Destination type. Use **`HTTP`** as a value for all HTTP(S) destinations.
`Authentication`	Authentication type. Use **`AppToAppSSO`** as a value.
`URL`	URL of the protected resource on the called application

Property	Description
`SessionCookieNames`	Optional. The *AppToApp* authentication module will use it to recognize the user session which improves the performance of the HTTP client. > **i Note** > In case that a session cookie name has a variable part you can specify it as a regular expression. You can specify more than one session cookie name as comma separated list: *Example*: **`JSESSIONID, JTENANTSESSIONID_.*, CookieName, Cookie*Name, CookieName.*`** > **i Note** > The spaces after comma are optional. If several cookies are listed, the session is recognized as soon as all of them are available in the response from the server. > **i Note** > Recommended value for the target Java app on SAP BTP is: **`JTENANTSESSIONID_.*`**, and for the HANA XS app is: **`xsId.*`**.
`saml2_audience`	Specifies a local SAML 2.0 provider name of the subaccount which consumes the target application. > **i Note** > If not specified, both applications must be consumed in the same subaccount.
`SkipSSOTokenGenerationWhenNoUser`	Optional. If this parameter is set and there is no user logged in, token generation is skipped, thus allowing anonymous access to public resources. If set, it may have any value.

Property	Description

`URL.headers.<header-key>`	Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a `URL.headers` prefix separated by dot-delimiter. For example:
	
	Sample Code
	`URL.headers.<header-key-1>=<header-value-1>`
	`...`
	`URL.headers.<header-key-n>=<header-value-n>`
	
	Note
	This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention. The HttpDestination (Version 2) [page 149] library supports this as of version 2.14.

Example

```
#
#Wed Jan 13 12:25:47 UTC 2016
Name=apptoapp
URL=https://someurl.com
ProxyType=Internet
Type=HTTP
SessionCookieNames=JTENANTSESSIONID_.*
Authentication=AppToAppSSO
```

Related Information

Server Certificate Authentication [page 93]
HTTP Proxy for On-Premise Connectivity [page 152]
AuthenticationHeaderProvider API [page 141]

1.4.3.2.6 Client Authentication Types for HTTP Destinations

Context

This section lists the supported client authentication types and the relevant supported properties.

No Authentication

This is used for destinations that refer to a service on the Internet or an on-premise system that does not require authentication. The relevant property value is:

```
Authentication=NoAuthentication
```

> **i** Note
>
> When a destination is using HTTPS protocol to connect to a Web resource, the JDK truststore is used as truststore for the destination.

Basic Authentication

This is used for destinations that refer to a service on the Internet or an on-premise system that requires basic authentication. The relevant property value is:

```
Authentication=BasicAuthentication
```

The following credentials need to be specified:

> **⚠** Caution
>
> Do not use your *own personal credentials* in the `<User>` and `<Password>` fields. Always use a *technical user* instead.

Property	Description
`User`	User name of the technical user to be used.

Property	Description
`Password`	Password of the technical user to be used.
`Preemptive`	If this property is not set or is set to **`TRUE`** (that is, the default behavior is to use preemptive sending), the authentication token is sent preemptively. Otherwise, it relies on the challenge from the server (401 HTTP code). The default value (used if no value is explicitly specified) is **`TRUE`**. For more information about preemptiveness, see http://tools.ietf.org/html/rfc2617#section-3.3 .

> **Note**
>
> When a destination is using the HTTPS protocol to connect to a Web resource, the JDK truststore is used as truststore for the destination.

> **Note**
>
> **`Basic Authentication`** and **`No Authentication`** can be used in combination with `ProxyType=`**`OnPremise`**. In this case, also the `CloudConnectorLocationId` property can be specified. As of Cloud Connector 2.9.0, you can connect multiple Cloud Connectors to a subaccount as long as their location ID is different. The value defines the location ID identifying the Cloud Connector over which the connection shall be opened. The default value is the empty string identifying the Cloud Connector that is connected without any location ID. This is also the case for all Cloud Connector versions prior to 2.9.0.

Client Certificate Authentication

This is used for destinations that refer to a service on the Internet. The relevant property value is:

`Authentication=`**`ClientCertificateAuthentication`**

The following credentials need to be specified:

Property	Description
`KeyStoreLocation` 1. When used in local environment 2. When used in cloud environment	Path to the JKS file that contains the client certificate(s) for authentication against a remote server. 1. The relative path to the JKS file. The root path is the server's location on the file system. 2. The name of the JKS file.
`KeyStorePassword`	The password for the key storage. This property is mandatory in case `KeyStoreLocation` is used.

> **Note**
>
> You can upload `KeyStore` JKS files using the same command for uploading destination configuration property file. You only need to specify the JKS file instead of the destination configuration file.

Configuration

- Configure Destinations from the Cockpit [page 77]
- Configure Destinations from the Eclipse IDE [page 64]
- Configure Destinations from the Console Client [page 56]

Related Information

Server Certificate Authentication [page 93]

1.4.3.2.7 OAuth Client Credentials Authentication

Create and configure an *OAuth2ClientCredentials* destination to consume OAuth-protected resources.

SAP BTP supports applications to use the OAuth client credentials flow for consuming OAuth-protected resources.

The client credentials are used to request an access token from an OAuth authorization server. If you use the HttpDestination API and DestinationFactory [page 136], the access token is automatically injected in all HTTP requests to the OAuth-protected resources. If you use the ConnectivityConfiguration API [page 139], you must retrieve the access token manually, using the AuthenticationHeaderProvider API [page 141] and inject it in the HTTP requests.

> **i Note**
>
> The retrieved access token is cached and auto-renovated. When a token is about to expire, a new token is created shortly before the expiration of the old one.

Configuration Steps

You can create and configure an *OAuth2ClientCredentials* destination using the properties listed below, and deploy it on SAP BTP. To create and configure a destination, follow the steps described in:

- Configure Destinations from the Cockpit [page 77], or
- Configure Destinations from the Console Client [page 56]

> **i Note**
>
> Configuring *OAuth2ClientCredentials* destinations from the Eclipse IDE is not yet supported.

Properties

The table below lists the destination properties required for the *OAuth2ClientCredentials* authentication type.

Property	Description
Required	
`Name`	Destination name. It must be the same as the destination name you use in your preferred configuration tool, that is, the console client or the *Destinations* editor (cockpit).
`Type`	Destination type. Use **`HTTP`** as value for all HTTP(S) destinations.
`URL`	URL of the protected resource on the called application.
`ProxyType`	You can use the proxy types **`Internet`** or **`OnPremise`**. If you use **`OnPremise`**, the OAuth server must be accessed through the Cloud Connector [page 232].
`Authentication`	Authentication type. Use **`OAuth2ClientCredentials`** as value.
`clientId`	Client ID used to retrieve the access token.
`clientSecret`	Client secret for the Client ID.
`tokenServiceURL`	URL of the OAuth server.
Optional	
`tokenServiceUser`	User for basic authentication to OAuth server (if required).
`tokenServicePassword`	Password for `tokenServiceUser` (if required).

Property	Description

`URL.headers.<header-key>`	Static key prefix used as a namespace grouping of the URL's HTTP headers whose values will be sent to the target endpoint. For each HTTP header's key, you must add a `URL.headers` prefix separated by dot-delimiter. For example:
	
	Sample Code
	`URL.headers.<header-key-1>=<header-value-1>`
	`...`
	`URL.headers.<header-key-n>=<header-value-n>`
	
	Note
	This is a naming convention. As the call to the target endpoint is performed on the client side, the service only provides the configured properties. The expectation for the client-side processing logic is to parse and use them. If you are using higher-level libraries and tools, please check if they support this convention. The HttpDestination (Version 2) [page 149] library supports this as of version 2.14.

Note

When the OAuth authorization server is called, it accepts the trust settings of the destination, see Server Certificate Authentication [page 93].

Example

Sample Code

```
URL=https://demo.sapjam.com/OData/OData.svc
Name=sap_jam_odata
TrustAll=true
ProxyType=Internet
Type=HTTP
Authentication=OAuth2ClientCredentials
tokenServiceURL=http://demo.sapjam.com/api/v1/auth/token
tokenServiceUser=tokenserviceuser
tokenServicePassword=pass
clientId=clientId
clientSecret=secret
```

1.4.3.3 RFC Destinations

RFC destinations provide the configuration needed for communicating with an on-premise ABAP system via RFC. The RFC destination data is used by the JCo version that is offered within SAP BTP to establish and manage the connection.

RFC Destination Properties

The RFC destination specific configuration in SAP BTP consists of properties arranged in groups, as described below. The supported set of properties is a subset of the standard JCo properties in arbitrary environments. The configuration data is divided into the following groups:

- User logon properties [page 113]
- Physical connection [page 118]
- Special parameters [page 122]
- Pooling configuration [page 115]
- Repository configuration [page 117]
- Principal Propagation SSO Authentication for RFC [page 124]

The minimal configuration contains user logon properties and information identifying the target host. This means you must provide at least a set of properties containing this information.

Example

```
Name=SalesSystem
Type=RFC
jco.client.client=000
jco.client.lang=EN
jco.client.user=consultant
jco.client.passwd=<password>
jco.client.ashost=sales-system.cloud
jco.client.sysnr=42
jco.destination.pool_capacity=5
jco.destination.peak_limit=10
```

1.4.3.3.1 User Logon Properties

JCo properties that cover different types of user credentials, as well as the ABAP system client and the logon language.

The currently supported logon mechanism uses user or password as the credentials.

Property	Description
`jco.client.client`	Represents the client to be used in the ABAP system. Valid format is a three-digit number.
`jco.client.lang`	Optional property. Represents the logon language. If the property is not provided, the user's or system's default language is used. Valid values are two-character ISO language codes or one-character SAP language codes.
`jco.client.user`	Represents the user to be used for logging on to the ABAP system. Max. 12 characters long. **i Note** When working with the *Destinations* editor in the cockpit, enter the value in the `<User>` field. Do not enter it as additional property.
`jco.client.alias_user`	Represents the user to be used for logging on to the ABAP system. Either `jco.client.user` or `jco.client.alias_user` must be specified. The alias user may be up to 40 characters long. **i Note** When working with the *Destinations* editor in the cockpit, enter the value in the `<Alias User>` field. Do not enter it as additional property.
`jco.client.passwd`	Represents the password of the user that is used. **i Note** Passwords in systems of SAP NetWeaver releases lower than 7.0 are case-insensitive and can be only eight characters long. For releases 7.0 and higher, passwords are case-sensitive with a maximum length of 40. **i Note** When working with the *Destinations* editor in the cockpit, enter this password in the `<Password>` field. Do not enter it as additional property.

Property	Description
`jco.client.tls_client_certificate_logon`	When set to `1`, the client certificate provided by the *KeyStore*, which must be configured in addition, is used for authentication instead of `jco.client.user`/`jco.client.alias_user` and `jco.client.passwd`. This property is only relevant for a connection using WebSocket RFC (`<Proxy Type>`=Internet). The default value is `0`. **i Note** When working with the Destinations editor in the cockpit, the `<User>`, `<Alias User>` and `<Password>` fields are hidden when setting the property to `1`. For more information on WebSocket RFC, see also: WebSocket RFC
`jco.destination.auth_type`	Optional property. - If the property is not provided, its default value **`CONFIGURED_USER`** is used, which means that user, password, or other credentials are specified directly. - To enable single sign-on via principal propagation (which means that the identity logged on in the cloud application is forwarded to the on-premise system), set the value to **`PrincipalPropagation`**. In this case, you do not need to provide `jco.client.user` and `jco.client.passwd` in the configuration. **i Note** For **`PrincipalPropagation`**, you should configure the properties `jco.destination.repository.user` and `jco.destination.repository.passwd` instead, since there are special permissions needed (for metadata lookup in the back end) that not all business application users might have.

1.4.3.3.2 Pooling Configuration

Learn about the JCo properties you can use to configure pooling in an RFC destination.

Overview

This group of JCo properties covers different settings for the behavior of the destination's connection pool. All properties are optional.

Property	Description
`jco.destination.pool_capacity`	Represents the maximum number of idle connections kept open by the destination. A value of **0** has the effect of no connection pooling, that is, connections will be closed after each request. The default value is **1**.
`jco.destination.peak_limit`	Represents the maximum number of active connections you can create for a destination simultaneously. Value **0** allows an unlimited number of active connections. Otherwise, if the value is less than the value of `jco.destination.pool_capacity`, it will be automatically increased to this value. Default setting is the value of `jco.destination.pool_capacity`. If `jco.destination.pool_capacity` is not specified, the default is **0** (unlimited).
`jco.destination.max_get_client_time`	Represents the maximum time in milliseconds to wait for a free connection in case the maximum number of active connections is already allocated by applications.
`jco.destination.expiration_time`	Represents the time in milliseconds after which idle connections that are available in the pool can be closed.
`jco.destination.expiration_check_period`	Represents the interval in milliseconds for the timeout checker thread to check the idle connections in the pool for expiration.
`jco.destination.pool_check_connection`	When setting this value to **1**, a pooled connection will be checked for corruption before being used for the next function module execution. Thus, it is possible to recognize corrupted connections and avoid exceptions being passed to applications when connectivity is basically working (default value is **0**). **i Note** Turning on this check has performance impact for stateless communication. This is due to an additional low-level ping to the server, which takes a certain amount of time for non-corrupted connections, depending on latency.

Pooling Details

- Each destination is associated with a connection factory and, if the pooling feature is used, with a connection pool.
- Initially, the destination's connection pool is empty, and the JCo runtime does not preallocate any connection. The first connection will be created when the first function module invocation is performed. The `peak_limit` property describes how many connections can be created simultaneously, if applications allocate connections in different sessions at the same time. A connection is allocated either when a stateless function call is executed, or when a connection for a stateful call sequence is reserved within a session.
- After the `<peak_limit>` number of connections has been allocated (in `<peak_limit>` number of sessions), the next session will wait for at most `<max_get_client_time>` milliseconds until a different session releases a connection (either finishes a stateless call or ends a stateful call sequence). In case the waiting session does not get any connection during the `<max_get_client_time>` period, the function request will be aborted with *JCoException* with the key **`JCO_ERROR_RESOURCE`**.
- Connections that are no longer used by applications are returned to the destination pool. There is at most a `<pool_capacity>` number of connections kept open by the pool. Further connections (`<peak_limit>` - `<pool_capacity>`) will be closed immediately after usage. The pooled connections (open connections in the pool) are marked as expired if they are not used again during `<expiration_time>` milliseconds. All expired connections will be closed by a timeout checker thread which executes the check every `<expiration_check_period>` milliseconds.

1.4.3.3.3 Repository Configuration

JCo properties that allow you to define the behavior of the repository that dynamically retrieves function module metadata.

All properties below are optional. Alternatively, you can create the metadata in the application code, using the metadata factory methods within the `JCo` class, to avoid additional round-trips to the on-premise system.

Property	Description
`jco.destination.repository_destination`	Specifies which destination should be used for repository queries. If the destination does not exist, an error occurs when trying to retrieve the repository. Defaults to itself.

Property	Description
`jco.destination.repository.user`	Optional property. If this property is set, and the repository destination is not set, it will be used as the user for repository queries. This configuration option allows using a different user for repository lookups with a single destination configuration, and restricting this user's permissions accordingly. See also SAP Note 460089. **Note** When working with the *Destinations* editor in the cockpit, enter the value in the `<Repository User>` field. Do not enter it as additional property.
`jco.destination.repository.passwd`	Represents the password for a repository user. If you use such a user, this property is mandatory. **Note** When working with the *Destinations* editor in the cockpit, enter this password in the `<Repository Password>` field. Do not enter it as additional property.

1.4.3.3.4 Target System Configuration

Learn about the JCo properties you can use to configure the target sytem information in an RFC destination (Neo environment).

Content

Overview [page 118]

Proxy Types [page 119]

Direct Connection [page 119]

Load Balancing Connection [page 120]

WebSocket Connection [page 121]

Overview

You can use the following configuration types alternatively:

- Direct connection to an ABAP application server
- Load balancing connection to a group of ABAP application servers via a message server
- WebSocket connection to an ABAP application server (RFC over Internet)

Depending on the configuration you use, different properties are mandatory or optional

Back to Content [page 118]

Proxy Types

The field `<Proxy Type>` lets you choose between `Internet` and `OnPremise`. When choosing `OnPremise`, the RFC communication is routed over a Cloud Connector that is connected to the subaccount. When choosing `Internet`, the RFC communciation is done over a WebSocket connection.

> **i Note**
>
> On SAP BTP trial (`hanatrial.ondemand.com`), you must add the parameter `jco.destination.proxy_type` with value **`OnPremise`** manually in section `<Additional Properties>`.

Back to Content [page 118]

Direct Connection

To use a direct connection to an application server over Cloud Connector, you must set the value for `<Proxy Type>` to `OnPremise`.

Property	Description
`jco.client.ashost`	Represents the application server host to be used. For configurations on SAP BTP, the property must match a virtual host entry in the Cloud Connector *Access Control* configuration. The property indicates that a direct connection is established.

Property	Description
`jco.client.sysnr`	Represents the so-called "system number" and has two digits. It identifies the logical port on which the application server is listening for incoming requests. For configurations on SAP BTP, the property must match a virtual port entry in the Cloud Connector *Access Control* configuration. **i Note** The virtual port in the above access control entry must be named **`sapgw<##>`**, where `<##>` is the value of `sysnr`.

Back to Content [page 118]

Load Balancing Connection

To use load balancing to a system over Cloud Connector, you must set the value for `<Proxy Type>` to `OnPremise`.

Property	Description
`jco.client.mshost`	Represents the message server host to be used. For configurations on SAP BTP, the property must match a virtual host entry in the Cloud Connector *Access Control* configuration. The property indicates that load balancing is used for establishing a connection.
`jco.client.group`	Optional property. Identifies the group of application servers that is used, the so-called "logon group". If the property is not specified, the group `PUBLIC` is used.
`jco.client.r3name`	Represents the three-character system ID of the ABAP system to be addressed. For configurations on SAP BTP, the property must match a virtual port entry in the Cloud Connector *Access Control* configuration. **i Note** The virtual port in the above access control entry must be named **`sapms<###>`**, where `<###>` is the value of `r3name`.

Property	Description
`jco.client.msserv`	Represents the port on which the message server is listening for incoming requests. you can use this property as an alternative to `jco.client.r3name`. One of these two must be present. For configurations on SAP BTP, the property must match a virtual port entry in the Cloud Connector *Access Control* configuration. You can therefore avoid lookups in the `/etc/services` file (`<Install_Drive>\Windows\System32\drivers\etc\services`) on the Cloud Connector host.

Back to Content [page 118]

WebSocket Connection

To use a direct connection over WebSocket, you must set the value for `<Proxy Type>` to `Internet`.

Prerequisites

- Your target system is an ABAP server as of S/4HANA (on-premise) version 1909.
- Your Java runtime version is at least one of the following:
  - **2.181.8** for *Java EE 6 Web Profile*
  - **2.146.17** for *Java Web Tomcat 7*
  - **3.101.18** for *Java Web Tomcat 8*
  - **1.70.18** for *Java EE 7 Web Profile TomEE 7*

Property	Description
`jco.client.wshost`	Represents the WebSocket RFC server host on which the target ABAP system is running. The system must be exposed to the Internet.
`jco.client.wsport`	Represents the WebSocket RFC server port on which the target ABAP system is listening.
`jco.client.tls_trust_all`	Optional property. If set to `1`, all server certificates are considered trusted during TLS handshake, if set to `0`, either a dedicated trust store must be configured, or the JDK trust store is used as default. Default value is `0`. **i Note** We recommend that you **do not use value 1 in productive scenarios**, but only for demo purposes.

Property	Description
`TrustStoreLocation` 1. When used in local environment 2. When used in cloud environment	If you don't want to use the standard JDK trust store as default (option *Use default JDK truststore* is unchecked), you must enter a `<Trust Store Location>`. This field indicates the path to the JKS file which contains trusted certificates (Certificate Authorities) for authentication against a remote client. 1. The relative path to the JKS file. The root path is the server's location on the file system. 2. The name of the JKS file. **i Note** The default JDK trust store is appended to the trust store defined in the destination configuration. As a result, the destination simultaneously uses both trust stores. If the `<Trust Store Location>` is not specified, the JDK trust store is used as default trust store for the destination.
`TrustStorePassword`	Password for the JKS trust store file. This property is mandatory if `<Trust Store Location>` is used.

i Note

You can upload trust store JKS files using the same command as for uploading destination configuration property files. You only need to specify the JKS file instead of the destination configuration file.

i Note

Connections to remote services which require *Java Cryptography Extension (JCE) unlimited strength jurisdiction policy* are not supported.

See also WebSocket RFC (ABAP Platform documentation).

Back to Content [page 118]

1.4.3.3.5 Parameters Influencing Communication Behavior

JCo properties that allow you to control the connection to an ABAP system.

All properties are optional.

Property	Description
`jco.client.trace`	Defines whether protocol traces are created. Valid values are **1** (trace is on) and **0** (trace is off). The default value is **0**.
`jco.client.codepage`	Declares the 4-digit SAP codepage that is used when initiating the connection to the backend. The default value is **1100** (comparable to iso-8859-1). It is important to provide this property if the password that is used contains characters that cannot be represented in **1100**.
`jco.client.delta`	Enables or disables table parameter delta management. It is enabled if set to **1**, and respectively disabled if set to **0**. The default value is **1**.
`jco.client.cloud_connector_version`	Defines the Cloud Connector version used for establishing a connection to the on-premise system. The default value is **2**. Currently, no other values are supported.
`jco.client.cloud_connector_location_id`	As of Cloud Connector 2.9.0, you can connect multiple Cloud Connectors to a subaccount as long as their location ID is different. The location ID specifies the Cloud Connector over which the connection is opened. The default value is an empty string identifying the Cloud Connector that is connected without any location ID. This is also valid for all Cloud Connector versions prior to 2.9.0. **i Note** When working with the *Destinations* editor in the cockpit, enter the Cloud Connector location ID in the `<Location ID>` field. Do not enter it as additional property.
`jco.client.serialization_format`	Defines the serialization format that is used when transferring function module data to the partner system. The property impacts the serialization behavior of function module data. Valid values are **columnBased** and **rowBased**. If you choose **columnBased**, the *fast RFC serialization* is used, as long as the partner system supports it, see SAP Note 2372888. When choosing the **rowBased** option, *classic* or *basXML* serialization are used. The default value is **rowBased**.
`jco.client.network`	Defines which network type is expected to be used for the destination.The property impacts the serialization behavior of function module data, see SAP Note 2372888. Valid values are **WAN** and **LAN**. The default value is **LAN**.

1.4.3.3.6 Principal Propagation SSO Authentication for RFC

Forward the identity of a cloud user from a Neo application to a backend system via RFC to enable single sign-on (SSO).

Context

A *PrincipalPropagation* destination enables single sign-on (SSO) by forwarding the identity of a cloud user to the Cloud Connector, and from there to the target on-premise system. In this way, the cloud user's identity can be provided without manual logon.

> **i** Note
>
> This authentication type applies only for on-premise connectivity.

Configuration Steps

You can create and configure a *PrincipalPropagation* destination by using the properties listed below, and deploy it on SAP BTP. For more information, see Managing Destinations.

Properties

The following credentials need to be specified:

Property	Description
`Name`	Destination name. Must be unique for the destination level.
`Type`	Destination type. Use **`RFC`** for all RFC destinations.
`jco.destination.auth_type`	Authentication type. Use **`PrincipalPropagation`** as value.
`jco.destination.proxy_type`	You can only use proxy type **`OnPremise`**.

Example

```
Name=SalesSystem
Type=RFC
jco.client.client=000
jco.destination.auth_type=PrincipalPropagation
jco.destination.proxy_type=OnPremise
```

```
jco.client.ashost=sales-system.cloud
jco.client.sysnr=42
```

1.4.3.4 LDAP Destinations

For your cloud applications, you can use LDAP-based user management if you are operating an LDAP server within your network.

LDAP destinations carry connectivity details for accessing systems over *Lightweight Directory Access Protocol* (LDAP) as specified in RFC 4511. In combination with the Cloud Connector they enable SAP BTP applications to access LDAP servers in an on-premise corporate network. LDAP destinations are intended to be used with the Java *JNDI/LDAP Service Provider*.

For more information on how to use the Java JNDI/LDAP Service Provider see: http://docs.oracle.com/javase/7/docs/technotes/guides/jndi/jndi-ldap.html.

Tasks

Task Type	Task
 Operator and/or Developer	Using LDAP Destination Properties [page 125]
	Configuring Connectivity for LDAP [page 126]
 Developer	Consuming Connectivity for LDAP [page 127]

Using LDAP Destination Properties

The following properties are predefined for LDAP destinations:

Predefined LDAP Destination Properties

Name (UI)	Name (internal)	Description/Values	Corresponding Java JNDI/ LDAP service provider property
Authentication	ldap.authentication	Possible values: *BasicAuthentication* or *NoAuthentication*	`java.naming.security.authentication` with value **`simple`** in case of *BasicAuthentication* and value **`none`** in case of *NoAuthentication*.
Password	ldap.password		`java.naming.security.credentials`
Proxy Type	ldap.proxyType	Possible values: *Internet* or *OnPremise*	In case proxy type is OnPremise, the resulting property is `java.naming.ldap.factory.socket` with value **`com.sap.core.connectivity.api.ldap.LdapOnPremiseSocketFactory`**.
URL	ldap.url	URL to the LDAP server. Must be in form: *ldap://<host>:<port>*. Example: *ldap://ldap-server.examplecompany.com:389*	`java.naming.provider.url`
User	ldap.user	Must be set in suitable form for the LDAP server. E.g. for Microsoft Active Directory the value should be in the following format: *<user_name>@<domain_name>*. Example: *serviceuser@examplecompany.com*	`java.naming.security.credentials`

As additional properties in an LDAP destination, you can specify the properties defined by the Java JNDI/LDAP Service Provider. For more details regarding these properties see *Environment Properties* at http://docs.oracle.com/javase/7/docs/technotes/guides/jndi/jndi-ldap.html I.

Back to Tasks [page 125]

Configuring Connectivity for LDAP

Configure an LDAP Destination for SAP BTP Applications



Configure Destinations from the Console Client [page 56]

Configure Destinations from the Eclipse IDE [page 64]

Configure Destinations from the Cockpit [page 77]

Configure Connectivity between an LDAP System and a Cloud Application



Configure the Cloud Connector in your on-premise network for LDAP:

Configuring the Cloud Connector for LDAP [page 288]

Back to Tasks [page 125]

Consuming Connectivity for LDAP



Consume the LDAP tunnel in a Java application, see Using LDAP [page 203].

Back to Tasks [page 125]

1.4.3.5 Mail Destinations

A mail destination is used to specify the mail server settings for sending or fetching e-mail, such as the e-mail provider, e-mail account, and protocol configuration.

The name of the mail destination must match the name used for the mail session resource. You can configure a mail destination directly in a destination editor or in a mail destination properties file. The mail destination then needs to be made available in the cloud. If a mail destination is updated, an application restart is required so that the new configuration becomes effective.

> **i** Note
>
> SAP does not act as e-mail provider. To use this service, please cooperate with an external e-mail provider of your choice.

Mail Destination Properties

The following properties are used to configure the mail destination:

Property	Description	Mandatory
`Name`	The name of the destination. The mail session that is configured by this mail destination is available by injecting the mail session resource `mail/<Name>`. The name of the mail session resource must match the destination name. The recommended name for a mail destination is `Session`.	Yes
`Type`	The type of destination. It must be `MAIL` for mail destinations.	Yes
`mail.*`	`javax.mail` properties for configuring the mail session. To send e-mails, you must specify at least `mail.transport.protocol` and `mail.smtp.host`. To retrieve e-mails, you must specify at least `mail.store.protocol`, `mail.<protocol>.host`, and for POP3 `mail.pop3.port`.	Depends on the mail protocol used.
`mail.password`	Password that is used for authentication. The user name for authentication is specified by `mail.user` (a standard `javax.mail` property).	Yes, if authentication is used (`mail.smtp.auth=true` and generally for fetching e-mail).

Note the following points:

- `mail.smtp.port`: The SMTP standard ports 465 (SMTPS) and 587 (SMTP+STARTTLS) are open for outgoing connections on SAP BTP.
- `mail.pop3.port`: The POP3 standard ports 995 (POP3S) and 110 (POP3+STARTTLS) are open for outgoing connections (used to fetch e-mail).
- `mail.imap.port`: The IMAP standard ports 993 (IMAPS) and 143 (IMAP +STARTTLS) are open for outgoing connections (used to fetch e-mail).
- `mail.<protocol>.host`: The mail server of an e-mail provider accessible on the Internet, such as Google Mail (for example, smtp.gmail.com, imap.gmail.com, and so on).

SMTP and IMAP Example

The destination below has been configured to use Gmail as the e-mail provider, SMTP with STARTTLS (port 587) for sending e-mail, and IMAP (SSL) for receiving e-mail:

```
Name=Session
Type=MAIL
mail.user=<gmail account name>
mail.password=<gmail account password>
mail.transport.protocol=smtp
mail.smtp.host=smtp.gmail.com
```

```
mail.smtp.auth=true
mail.smtp.starttls.enable=true
mail.smtp.port=587
mail.store.protocol=imaps
mail.imaps.host=imap.gmail.com
```

SMTPS Example

The destination below uses Gmail and SMTPS (port 465) for sending e-mail:

```
Name=Session
Type=MAIL
mail.user=<gmail account name>
mail.password=<gmail account password>
mail.transport.protocol=smtps
mail.smtps.host=smtp.gmail.com
mail.smtps.auth=true
mail.smtps.port=465
```

Related Information

JavaMail API Documentation
Configure Destinations from the Eclipse IDE [page 64]
Configure Destinations from the Cockpit [page 77]
Configure Destinations from the Console Client [page 56]

1.4.3.6 Principal Propagation

Forward the identity of cloud users to an on-premise system to enable single sign-on (Neo environment).

Content

Overview [page 130]
How It Works [page 130]
Using the Principal Propagation Property in Destinations [page 131]
Tasks [page 131]

Overview

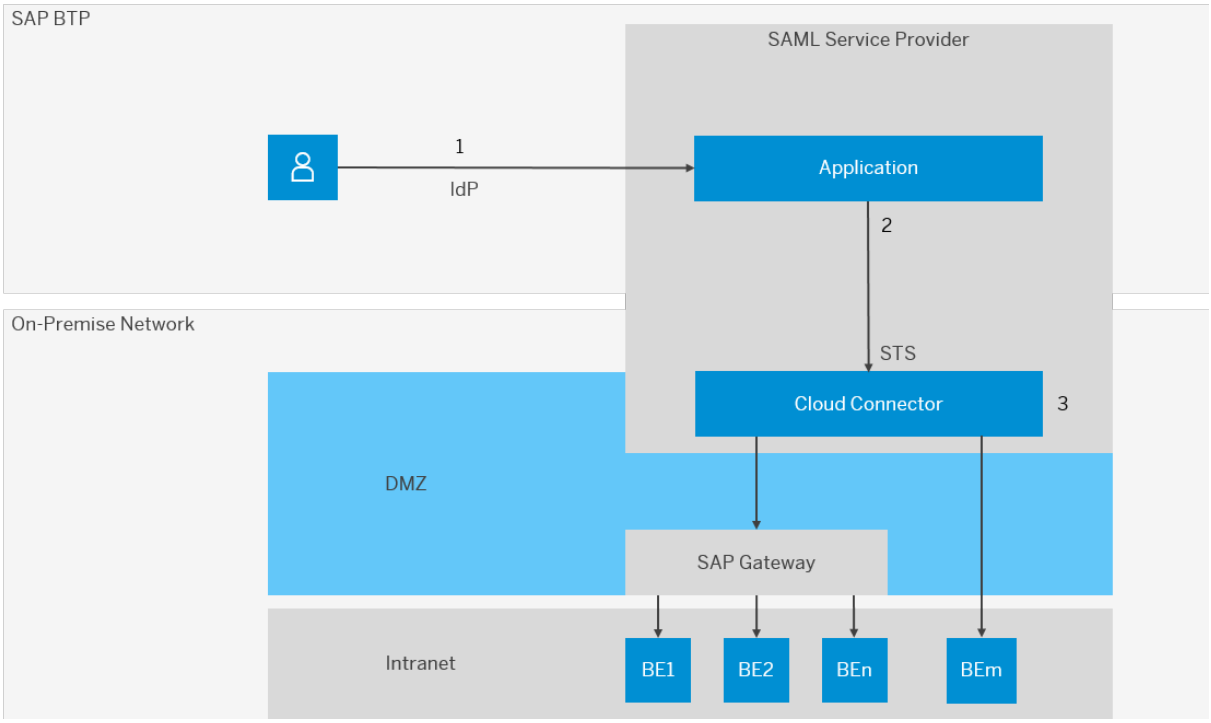
The Connectivity service provides a secure way of forwarding the identity of a cloud user to the Cloud Connector, and from there to an on-premise system. This process is called **principal propagation**.

It uses a SAML token as exchange format for the user information. User mapping is done in the back end. The token is forwarded either directly, or an `X.509` certificate is generated, which is then used in the backend.

> **!** Restriction
>
> This authentication is only applicable if you connect to your on-premise system via the Cloud Connector.

Back to Content [page 129]

How It Works



1. The user authenticates at the cloud application front end via the IdP (Identity Provider) using a standard SAML Web SSO profile. When the backend connection is established by the cloud application, the destination service (re)uses the received SAML assertion to create the connection to the on-premise backend system (BE1-BEm).
2. The Cloud Connector validates the received SAML assertion for a second time, extracts the attributes, and uses its STS (Security Token Service) component to issue a new token (an `X.509` certificate) with the same or similar attributes to assert the identity to the backend.
3. The Cloud Connector and the cloud application share the same SAML service provider identity, which means that the trust is only set up once in the IdP.

Back to Content [page 129]

Using the Principal Propagation Property in Destinations

You can create and configure connectivity destinations using the `PrincipalPropagation` property in the Eclipse IDE and in the cockpit. Keep in mind that this property is only available for destination configurations created in the cloud.

- Create and Delete Destinations on the Cloud [page 69] (Eclipse IDE, procedure and examples)
- Create Destinations (Cockpit) [page 79] (procedure and examples)

Back to Content [page 129]

Tasks

- Configuring Principal Propagation [page 301] (Cloud Connector)
- Principal Propagation SSO Authentication for HTTP [page 99] (SAP BTP destination)
- Principal Propagation SSO Authentication for RFC [page 124] (SAP BTP destination)

Back to Content [page 129]

Related Information

Authentication Configuration [page 1742]

1.4.3.7 Multitenancy in the Connectivity Service

Using multitenancy for applications that require a connection to a remote service or on-premise application.

Endpoint Configuration

Applications that require a connection to a remote service can use the Connectivity service to configure HTTP or RFC endpoints. In a provider-managed application, such an endpoint can either be once defined by the application provider (Provider-Specific Destination [page 132]), or by each application consumer (Consumer-Specific Destination [page 133]).

If the application needs to use the same endpoint, independently from the current application consumer, the destination that contains the endpoint configuration is uploaded by the application provider. If the endpoint should be different for each application consumer, the destination can be uploaded by each particular application consumer.

To prevent application consumers from using an individual endpoint for a provider application, you can set the property `DestinationProvider=Application` in the HTTP or RFC destination. In this case, the destination is always read from the provider application.

> **i Note**
>
> This connectivity type is fully applicable also for on-demand to on-premise connectivity.

Destination Levels

You can configure destinations simultaneously on three levels: *subscription*, *consumer subaccount* and *application*. This means that it is possible to have one and the same destination on more than one configuration level. For more information, see Managing Destinations [page 54].

Destination visibility according to the level, when uploaded on:

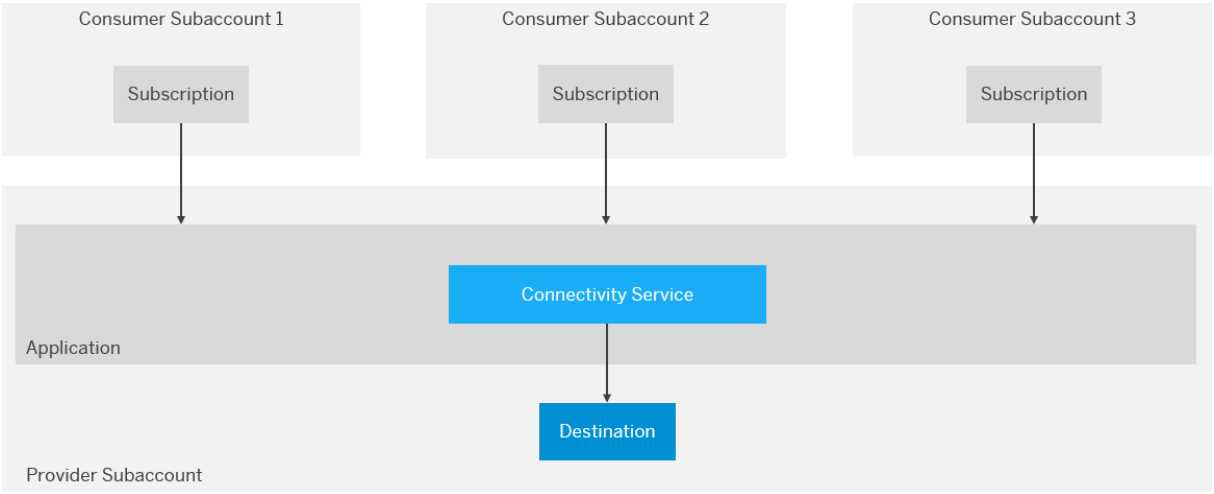
Level	Visibility
Subscription level	Only visible for the dedicated subscription.
Subaccount level	Visible for the whole consumer subaccount.
Application level	Visible by all tenants and subaccounts, regardless their permission settings.

When the application accesses the destination at runtime, the Connectivity service

1. looks up the requested destination in the consumer subaccount on *subscription* level. If no destination is available there, it
2. checks if the destination is available on the *subaccount* level of the consumer subaccount. If there is still no destination found, it
3. searches on *application* level of the provider subaccount.

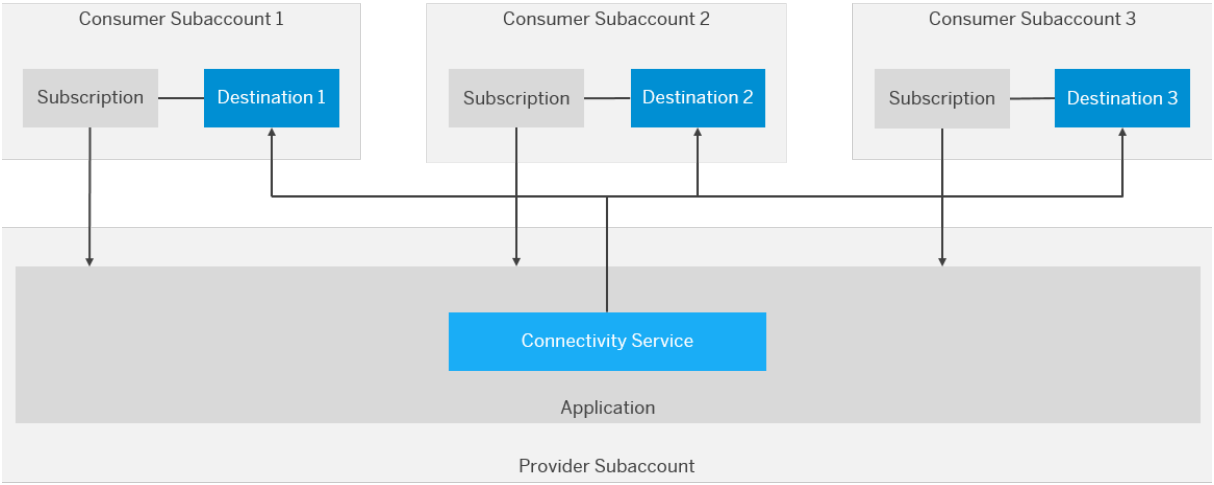
Back to Top [page 131]

Provider-Specific Destination



Back to Top [page 131]

Consumer-Specific Destination



Back to Top [page 131]

Related Information

Developing Multitenant Applications in the Neo Environment [page 898]
Create a Multitenant Connectivity Application [page 911]

1.4.3.8 Configuring Backup

Find an overview of backup procedures for your destination configurations.

Export Destinations (Cockpit) [page 88]	Export destinations from the cockpit.
Export Destinations (IDE) [page 73]	Export destinations from the IDE.
Download Destinations [page 60]	Download destinations from the console client.

1.4.4 Development

Consume the Connectivity service from a Java or HANA XS application in the Neo environment and use the Destination Configuration service to provide the required destination information.

Task	Description
Consuming the Connectivity Service (Java) [page 134]	Connect your Java cloud applications to the Internet, make cloud-to-on-premise connections to SAP or non-SAP systems, or send and fetch e-mail.
Consuming the Connectivity Service (HANA XS) [page 216]	Create connectivity destinations for HANA XS applications, configure security, add roles and test them in an enterprise or trial landscape.
Consuming the Destination Configuration Service [page 230]	Retrieve destination configurations for your cloud application in the Neo environment, in a secure and reliable way.

1.4.4.1 Consuming the Connectivity Service (Java)

Connect your Java cloud applications to the Internet, make cloud-to-on-premise connections to SAP or non-SAP systems, or send and fetch e-mail.

Task	Description
Connectivity and Destination APIs [page 135]	Find an overview of the available connectivity and destination APIs.
Exchanging Data via HTTP [page 146]	Consume the Connectivity service using the HTTP protocol.
Invoking ABAP Function Modules via RFC [page 191]	Call a remote-enabled function module in an ABAP server using the SAP Java Connector (JCo) API.
Using LDAP [page 203]	You can use LDAP-based user management if you are operating an LDAP server within your local network.

Task	Description
Using the TCP Protocol for Cloud Applications [page 204]	Access on-premise systems via TCP-based protocols, using a SOCKS5 proxy.
Sending and Fetching E-Mail [page 208]	Send mail messages from your cloud applications using e-mail providers that are accessible on the Internet.

1.4.4.1.1 Connectivity and Destination APIs

Destinations

Destinations are part of SAP Connectivity service and are used for the outbound communication from a cloud application to a remote system. They contain the connection details for the remote communication of an application, which can be configured for each customer to accommodate the specific customer back-end systems and authentication requirements. For more information, see Managing Destinations [page 54].

Destinations should be used by application developers when they aim to provide applications that:

- Integrate with remote services or back-end systems that need to be configured by customers
- Integrate with remote services or back-end systems that are located in a fenced environment (that is, behind firewalls and not publicly accessible)

> → Tip
>
> HTTP clients created by destination APIs allow parallel usage of HTTP client instances (via class `ThreadSafeClientConnManager`).

Connectivity APIs

Package	Description
`org.apache.http`	http://hc.apache.org
`org.apache.http.client`	http://hc.apache.org/httpcomponents-client-ga/httpclient/apidocs/org/apache/http/client/package-summary.html
`org.apache.http.util`	http://hc.apache.org/httpcomponents-core-ga/httpcore/apidocs/org/apache/http/util/package-summary.html
`javax.mail`	https://javamail.java.net/nonav/docs/api/

Package	Description
	The SAP BTP SDK for Java Web uses version 1.4.1 of `javax.mail`, the SDK for Java EE 6 Web Profile uses version 1.4.5 of `javax.mail`, and the SDK for Java Web Tomcat 7 uses version 1.4.7 of `javax.mail`.
`com.sap.core.connectivity.api`	You can find the Connectivity service API in directory `<SDK_location>/javadoc/com/sap/core/connectivity/api>`. For more information, see API Documentation [page 1147].

Destination APIs

- JavaMail API [page 209]
- HttpDestination Library [page 147]
- Java Connector API [page 141]
- ConnectivityConfiguration API [page 139]
- AuthenticationHeaderProvider API [page 141]
- Principal Propagation Using HTTP Proxy [page 154]
- HttpDestination API and DestinationFactory [page 136]

Destination Configuration Tasks

- Configure Destinations from the Eclipse IDE [page 64]
- Configure Destinations from the Console Client [page 56]
- Configure Destinations from the Cockpit [page 77]
- Consuming the Destination Configuration Service [page 230]

1.4.4.1.1.1 HttpDestination API and DestinationFactory

All connectivity API packages are visible by default from all Web applications. Applications can consume the destinations via a JNDI lookup.

Procedure

Prerequisites

You have set up your Java development environment. See also: Setting Up the Development Environment [page 845]

Retrieving HTTP Destinations Using HttpDestination API

To consume destinations using `HttpDestination` API, you need to define your destination as a resource in the `web.xml` file.

1. An example of a destination resource named **`myBackend`**, which is described in the `web.xml` file, is as follows:

```
<resource-ref>
    <res-ref-name>myBackend</res-ref-name>
    <res-type>com.sap.core.connectivity.api.http.HttpDestination</res-type>
</resource-ref>
```

2. In your servlet code, you can look up the destination (an HTTP destination in this example) from the JNDI registry as following:

```
import javax.naming.Context;
import javax.naming.InitialContext;
import com.sap.core.connectivity.api.http.HttpDestination;
...

// coding to lookup the destination "myBackend"
Context ctx = new InitialContext();
HttpDestination destination = (HttpDestination) ctx.lookup("java:comp/env/
myBackend");
```

> **i Note**
>
> If you want the lookup name to differ from the destination name, you can specify the lookup name in `<res-ref-name>` and the destination name in `<mapped-name>`, as shown in the following example:

```
<resource-ref>
   <res-ref-name>myLookupName</res-ref-name>
   <res-type>com.sap.core.connectivity.api.http.HttpDestination</res-type>
   <mapped-name>myBackend</mapped-name>
</resource-ref>
```

3. With the retrieved HTTP destination, you can then, for example, send a simple GET request to the configured remote system by using the following code:

```
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.HttpResponse;
...

// coding to call service "myService" on the system configured in the given
destination
HttpClient createHttpClient = destination.createHttpClient();
HttpGet get = new HttpGet("myService");
HttpResponse resp = createHttpClient.execute(get);
```

> **Note**
>
> If you want to use `<res-ref-name>`, which contains "/", the name after the last "/" should be the same as the destination name. For example, you can use `<res-ref-name>connectivity/myBackend</res-ref-name>`. In this case, you should use `java:comp/env/connectivity/myBackend` as a lookup string.

If you want to get the URL of your configured destination, use the `URI getURI()` method. This method returns the URL, defined in the destination configuration, converted to URI.

Retrieving HTTP Destinations Using DestinationFactory

As alternative approach how to retrieve an HTTP destination, `DestinationFactory` can be used. We recommend this approach if the used destinations are unknown at implementation time and should be loaded dynamically at runtime.

1. Define the `DestinationFactory` as a JNDI resource in the `web.xml` file:

```
<resource-ref>
   <res-ref-name>connectivity/DestinationFactory</res-ref-name>
   <res-type>com.sap.core.connectivity.api.DestinationFactory</res-type>
</resource-ref>
```

2. In your Java code, you can then look it up and use it in following way:

```
DestinationFactory destinationFactory = (DestinationFactory)
ctx.lookup(DestinationFactory.JNDI_NAME);
   destination = (HttpDestination)
destinationFactory.getDestination(destinationName);
```

> **Note**
>
> If you have two destinations with the same name, one configured on subaccount level and the other on application level, the `getConfiguration()` method will return the destination on subaccount level.
>
> The preference order is: subscription level -> subaccount level -> application level.

Related Information

If you need to also add Maven dependencies, take a look at this blog:

Building Java Web Applications with Maven

See also:

Maven Plugin [page 815]

1.4.4.1.1.2 ConnectivityConfiguration API

All connectivity API packages are visible by default from all Web applications. Applications can consume the connectivity configuration via a JNDI lookup.

Context

Besides making destination configurations, you can also allow your applications to use their own HTTP clients. The `ConnectivityConfiguration` API provides you a direct access to the destination configurations of your applications. This API also:

- Can be used independent of the existing destination API so that applications can bring and use their own HTTP client
- Consists of both a **`public REST API`** and a **`Java client API`**.

The `ConnectivityConfiguration` API is supported by all runtimes, including **`Java Web Tomcat 7`**. For more information about runtimes, see Application Runtime Container [page 857].

To learn how to retrieve destination configurations, follow the procedure below.

Procedure

1. To consume connectivity configuration using JNDI, you need to define `ConnectivityConfiguration` API as a resource in the `web.xml` file. An example of a `ConnectivityConfiguration` resource named **`connectivityConfiguration`**, which is described in the `web.xml` file, is as follows:

   ```
   <resource-ref>
       <res-ref-name>connectivityConfiguration</res-ref-name>
       <res-
   type>com.sap.core.connectivity.api.configuration.ConnectivityConfiguration</
   res-type>
   </resource-ref>
   ```

2. In your servlet code, you can look up the `ConnectivityConfiguration` API from the JNDI registry as following:

   ```
   import javax.naming.Context;
   import javax.naming.InitialContext;
   import com.sap.core.connectivity.api.configuration.ConnectivityConfiguration;
   ...

   // look up the connectivity configuration API "connectivityConfiguration"
   Context ctx = new InitialContext();
   ConnectivityConfiguration configuration = (ConnectivityConfiguration)
   ctx.lookup("java:comp/env/connectivityConfiguration");
   ```

3. With the retrieved `ConnectivityConfiguration` API, you can read all properties of any destination defined on subscription, application or subaccount level.

> ### Note
>
> If you have two destinations with the same name, one configured on subaccount level and the other on application level, the `getConfiguration()` method will return the destination on subaccount level. The preference order is: subscription level -> subaccount level -> application level.

```
// get destination configuration for "myDestinationName"
DestinationConfiguration destConfiguration =
configuration.getConfiguration("myDestinationName");
// get the "myDestinationName" authentication property (example)
String value = destConfiguration.getProperty("Authentication");
// get all destination properties
Map<String, String> allDestinationPropeties =
destConfiguration.getAllProperties();
```

4. If truststore and keystore are defined in the corresponding destination, they can be accessed by using methods `getKeyStore` and `getTrustStore`.

```
// get destination configuration for "myDestinationName"
DestinationConfiguration destConfiguration =
configuration.getConfiguration("myDestinationName");

// get the configured keystore
KeyStore keyStore = destConfiguration.getKeyStore();

// get the configured truststore
KeyStore trustStore = destConfiguration.getTrustStore();

// create sslcontext
TrustManagerFactory tmf =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
tmf.init(trustStore);

KeyManagerFactory keyManagerFactory =
KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
// get key store password from destination
String keyStorePassword = destConfiguration.getProperty("KeyStorePassword");
keyManagerFactory.init(keyStore, keyStorePassword.toCharArray());

SSLContext sslcontext = SSLContext.getInstance("TLS");
sslcontext.init(keyManagerFactory.getKeyManagers(), tmf.getTrustManagers(),
null);
SSLSocketFactory sslSocketFactory = sslcontext.getSocketFactory();

// get the destination URL
String value = destConfiguration.getProperty("URL");
URL url = new URL(value);

// use the sslcontext for url connection
URLConnection urlConnection = url.openConnection();
((HttpsURLConnection) urlConnection).setSSLSocketFactory(sslSocketFactory);
urlConnection.connect();
InputStream in = urlConnection.getInputStream();
...
```

1.4.4.1.1.3 Java Connector API

The Java Connector (JCo) is a middleware component that enables you to develop ABAP-compliant components and applications in Java.

JCo supports communication with Application Server ABAP (AS ABAP) in both directions:

- Inbound - Java calls ABAP
- Outbound - ABAP calls Java

The JCo can be implemented with desktop applications and Web server applications.

> **i Note**
>
> You can find generic information regarding authorizations required for the use of JCo in SAP Note 460089.

To learn in detail about the JCo API, see the JCo 3.0 documentation on SAP Support Portal.

> **i Note**
>
> This documentation contains sections not applicable to SAP BTP. In particular:
>
> - **Architecture**: CPIC is only used in the last mile from your Cloud Connector to the backend. From the cloud to the Cloud Connector, SSL protected communication is used.
> - **Installation**: SAP BTP already includes all the necessary artifacts.
> - **Customizing and Integration**: In SAP BTP, the integration is already done by the runtime. You can concentrate on your business application logic.
> - **Server Programming**: The programming model of JCo in SAP BTP does not include server-side RFC communication.
> - **IDoc Support for External Java Applications**: For the time being, there is no IDocLibrary for JCo available in SAP BTP.

Related Information

Invoking ABAP Function Modules via RFC [page 191]

1.4.4.1.1.4 AuthenticationHeaderProvider API

Implement authentication token generation for your Web application using the `AuthenticationHeaderProvider` API.

Context

The `AuthenticationHeaderProvider` API allows your Web applications to use their own HTTP clients, providing authentication token generation for application-to-application SSO (single sign-on) and on-premise SSO.

This API:

- Provides additional helper methods, which facilitate the task to initialize an HTTP client (for example, an authentication method that helps you set headers for application-to-application SSO).
- Consists of both a public REST API and a Java client API. See also API Documentation [page 1147].

All connectivity API packages are visible by default from all Web applications. Applications can consume the authentication header provider via a JNDI lookup.

> **i Note**
>
> The `AuthenticationHeaderProvider` API is supported by all runtimes, including **`Java Web Tomcat 7`**. For more information about runtimes, see Application Runtime Container [page 857].

Tasks

Retrieve Authentication Header Providers [page 142]

Generate Application-to-Application SSO Authentication [page 143]

Generate On-Premise SSO Authentication [page 144]

Generate SAPAssertionSSO Headers [page 144]

Generate OAuth2SAMLBearerAssertion Headers [page 144]

Generate Client Credentials Headers [page 145]

Back to Context [page 142]

Retrieve Authentication Header Providers

1. To consume the `AuthenticationHeaderProvider` API using JNDI, you need to define it as a resource in the `web.xml` file. An example of an `AuthenticationHeaderProvider` resource named **`myAuthHeaderProvider`**, which is described in the `web.xml` file, looks like this:

```
<resource-ref>
    <res-ref-name>myAuthHeaderProvider</res-ref-name>
    <res-
type>com.sap.core.connectivity.api.authentication.AuthenticationHeaderProvider
</res-type>
```

```
</resource-ref>
```

2. In your servlet code, you can look up the `AuthenticationHeaderProvider` API from the JNDI registry:

```
import javax.naming.Context;
import javax.naming.InitialContext;
import
com.sap.core.connectivity.api.authentication.AuthenticationHeaderProvider;
...

// look up the connectivity authentication header provider resource called
"myAuthHeaderProvider"
Context ctx = new InitialContext();
AuthenticationHeaderProvider authHeaderProvider =
(AuthenticationHeaderProvider) ctx.lookup("java:comp/env/
myAuthHeaderProvider");
```

Back to Tasks [page 142]

Back to Context [page 142]

Generate Application-to-Application SSO Authentication

The `AuthenticationHeaderProvider` API can generate an authorization header to be used in a scenario of application-to-application communication where the caller needs to propagate its logged-in user. Both applications are deployed on SAP BTP and consumed within a single subaccount. The header must be embedded in the request to the target application.

> → Tip
>
> We recommend that you pack the HTTP client (Apache or other) inside the `lib` folder of your Web application archive.

Prerequisites:

- Principal propagation must be enabled for the subaccount. For more information, see Application Identity Provider [page 1720] → section *Specifying Custom Local Provider Settings*.
- Both applications must run on behalf of the same subaccount.
- The receiving application must use SAML2 authentication.

> i Note
>
> If you work with the **`Java Web Tomcat 7`** runtime, bear in mind that the following code snippet works properly only when using the Apache HTTP client version **4.1.3**. If you use other (higher) versions of the Apache HTTP client, you should adapt your code.

```
// retrieve the authorization header for application-to-application SSO
AuthenticationHeader appToAppSSOHeader =
authHeaderProvider.getAppToAppSSOHeader(url);


// create an HTTP client and add the header to the request
HttpClient httpClient = new DefaultHttpClient();
```

```
HttpGet request = new HttpGet(url);
request.addHeader(appToAppSSOHeader.getName(), appToAppSSOHeader.getValue());

// execute the request
HttpResponse response = httpClient.execute(request);
```

Back to Tasks [page 142]

Back to Context [page 142]

Generate On-Premise SSO Authentication

To learn how to generate on-premise SSO authentication, see Principal Propagation Using HTTP Proxy [page 154].

Back to Tasks [page 142]

Back to Context [page 142]

Generate SAPAssertionSSO Headers

The aim of the SAPAssertionSSO headers is to generate an assertion ticket that propagates the currently logged-in SAP BTP user to an SAP back-end system. You can use this authentication type only if the user IDs on both sides are the same.

The `AuthenticationHeaderProvider` API provides the following method for generating such headers:

```
AuthenticationHeader getSAPAssertionHeader(DestinationConfiguration
destinationConfiguration);
```

See also: SAP Assertion SSO Authentication [page 96].

Back to Tasks [page 142]

Back to Context [page 142]

Generate OAuth2SAMLBearerAssertion Headers

SAP BTP supports applications to use the SAML Bearer assertion flow for consuming OAuth-protected resources. As a result, applications do not need to deal with some of the complexities of OAuth and can reuse existing identity providers for user data. Users are authenticated by using SAML against the configured trusted identity providers. The SAML assertion is then used to request an access token from an OAuth authorization server. This access token should be injected in all HTTP requests to the OAuth-protected resources.

> **Note**
>
> The access tokens are cached by the `AuthenticationHeaderProvider` and are auto-renovated. When a token is about to expire, a new token is created shortly before the expiration of the old one.

The `AuthenticationHeaderProvider` API provides the following method for generating such headers:

```
List<AuthenticationHeader>
getOAuth2SAMLBearerAssertionHeaders(DestinationConfiguration
destinationConfiguration);
```

See also: SAML Bearer Assertion Authentication [page 101].

Back to Tasks [page 142]

Back to Context [page 142]

Generate Client Credentials Headers

SAP BTP also supports applications to use the OAuth client credentials flow for consuming OAuth-protected resources.

You can use the client credentials to request an access token from an OAuth authorization server. If you use the HttpDestination API and DestinationFactory [page 136], the access token is automatically injected in all HTTP requests to the OAuth-protected resources. If you use the ConnectivityConfiguration API [page 139], you must retrieve the access token using the `AuthenticationHeaderProvider` API and manually inject it in the HTTP requests.

> **Note**
>
> The access tokens are cached by the `AuthenticationHeaderProvider` and are auto-renovated. When a token is about to expire, a new token is created shortly before the expiration of the old one.

The `AuthenticationHeaderProvider` API provides the following method for generating such headers:

```
AuthenticationHeader getOAuth2ClientCredentialsHeader (DestinationConfiguration
destinationConfiguration);
```

See also: OAuth Client Credentials Authentication [page 110].

Back to Tasks [page 142]

Back to Context [page 142]

Related Information

HTTP Proxy for On-Premise Connectivity [page 152]

1.4.4.1.2 Exchanging Data via HTTP

Consuming Connectivity via HTTP

- Call an Internet service using a simple application that queries some information from a public service:
  Consume Internet Services (Java Web or Java EE 6 Web Profile) [page 155]
  Consume Internet Services (Java Web Tomcat 7) [page 163]
- Call a service from a fenced customer network using a simple application that consumes an on-premise ping service:
  Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170]
  Consume Backend Systems (Java Web Tomcat 7) [page 181]

Configuring Connectivity via HTTP

- Configure an application using destinations:
  Configure Destinations from the Eclipse IDE [page 64]
  Configure Destinations from the Console Client [page 56]
  Configure Destinations from the Cockpit [page 77]
- Configure connectivity between a customer system and an on-demand application. You must install the Cloud Connector in your internal network and then configure it to expose an on-premise service. For more information, see Initial Configuration (HTTP) [page 285] and Configure Access Control (HTTP) [page 331].

Connecting to On-Premise Back-End Services

You can consume on-premise back-end services in two ways – via HTTP destinations and via the HTTP Proxy. For more information, see:

- HTTP Destinations [page 91]
- HTTP Proxy for On-Premise Connectivity [page 152]

Connecting to a Local Host

To create a loopback connection, you can use the dedicated HTTP port bound to localhost. The port number can be obtained from the cloud environment variable `HC_LOCAL_HTTP_PORT`.

For more information, see Using Cloud Environment Variables [page 873] → section "List of Environment Variables".

> **Note**
>
> Note that when deploying locally from the Eclipse IDE or the console client, the HTTP port may differ.

Related Information

Using the Keystore Service for Client Side HTTPS Connections [page 1791]

1.4.4.1.2.1 HttpDestination Library

Learn about the different `HttpDestination` library versions available for the Connectivity service.

`HttpDestination` is a Java library that simplifies the consumption of destination configurations and provides smooth integration with on-premise systems for SAP BTP services like Security and Connectivity.

There are two major versions of the library:

- HttpDestination (Version 1) [page 147]: Available in the Neo SDK, package `<SDK_location>/javadoc/com/sap/core/connectivity/api`.

  > **Caution**
  >
  > `HttpDestination` v1 is a deprecated legacy version. We recommend that you use `HttpDestination` v2.

- HttpDestination (Version 2) [page 149]: based on Apache HttpClient 4.5, available in the Maven Central Repository .

Related Information

Connectivity and Destination APIs [page 135]

1.4.4.1.2.1.1 HttpDestination (Version 1)

Learn about the legacy `HttpDestination` library version available for the Connectivity service.

Overview

By default, all Connectivity API packages are visible from all Web applications. In this classical case, applications can consume the destinations via a JNDI lookup. For more information, see Connectivity and Destination APIs [page 135].

There are specific cases though, when the destination names are not known in advance and cannot be defined in the `web.xml` file. This is relevant to HTTP destinations, and in these cases, you must use the `DestinationFactory` JNDI lookup (`com.sap.core.connectivity.api.DestinationFactory`). To do this, follow the procedure below.

> ⚠ Caution
>
> - If you use the SDK for Java Web, we only *recommend* that you create a destination before deploying the application.
> - If you use the SDK for Java EE 6 Web Profile, you *must* create a destination before deploying the application.
> - If you use the SDK for Java Web Tomcat 7, the `DestinationFactory` API is not supported. Instead, you can use the ConnectivityConfiguration API [page 139].
> - If yue use the SDK for Java Web Tomcat 8, you can use the ConnectivityConfiguration API [page 139]. We recommend that you use HttpDestination (Version 2) [page 149].

> → Tip
>
> If you know in advance the names of all destinations you need, you should rather use destinations. Otherwise, we recommend that you use `DestinationFactory`.

Procedure

To look up the destination factory using JNDI, follow these steps:

1. For your application, define a reference in the `web.xml` file:

   > ‹≡› Sample Code

   ```
   <resource-ref>
       <res-ref-name>connectivity/DestinationFactory</res-ref-name>
       <res-type>com.sap.core.connectivity.api.DestinationFactory</res-type>
   </resource-ref>
   ```

2. Use the following code to look up the `DestinationFactory`:

   > ‹≡› Sample Code

   ```
   import com.sap.core.connectivity.api.DestinationFactory;
   import com.sap.core.connectivity.api.http.HttpDestination
   ```

```
...
Context ctx = new InitialContext();
DestinationFactory destinationFactory
=(DestinationFactory)ctx.lookup(DestinationFactory.JNDI_NAME);
HttpDestination destination = (HttpDestination)
destinationFactory.getDestination("myBackend");
```

3. With the retrieved HTTP destination, you can then, for example, send a simple *GET* request to the configured remote system by using the following code:

Sample Code

```
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.HttpResponse;
...

// coding to call service "myService" on the system configured in the
given destination
HttpClient createHttpClient = destination.createHttpClient();
HttpGet get = new HttpGet("myService");
HttpResponse resp = createHttpClient.execute(get);
```

1.4.4.1.2.1.2 HttpDestination (Version 2)

Use the current `HttpDestination` library version available for the Connectivity service.

The `HttpDestination` library is released to the Maven central repository and can be used from there . This is the recommended way to consume `HttpDestination` in the Neo environment.

Since on the Java Web Tomcat 8 runtime the `HttpDestination` functionality from Java Web is not part of the runtime, you must package some libraries with your application to use this functionality.

For **Maven** projects, you can easily achieve this by using the `pom.xml` file.

Reference HttpDestination

> ⚠ Caution
>
> You must add both the `HttpDestination` dependencies and the external dependencies.

Add the `HttpDestination` library as a dependency in your `pom.xml` file as described in the Maven Central Repository .

Sample Code

```
<dependency>
    <groupId>com.sap.cloud.connectivity</groupId>
    <artifactId>sap-cloud-connectivity-httpdestination</artifactId>
    <version>2.11.0</version>
</dependency>
```

External Dependencies

Some external dependencies are also required. Add the following dependencies to your `pom.xml` file.

Caution

Once you put those jars in your `web-inf/lib`, your application will **no longer work** on runtime 1 (OSGi runtime).

Sample Code

```
<dependency>
   <groupId>org.apache.httpcomponents</groupId>
   <artifactId>httpclient</artifactId>
   <version>4.5.10</version>
</dependency>
<dependency>
   <groupId>org.apache.httpcomponents</groupId>
   <artifactId>httpcore</artifactId>
   <version>4.4.12</version>
</dependency>
<dependency>
   <groupId>org.apache.httpcomponents</groupId>
   <artifactId>httpmime</artifactId>
   <version>4.5.2</version>
</dependency>
<dependency>
   <groupId>org.apache.httpcomponents</groupId>
   <artifactId>httpclient-cache</artifactId>
   <version>4.5.3</version>
</dependency>
<dependency>
   <groupId>commons-codec</groupId>
   <artifactId>commons-codec</artifactId>
   <version>1.13</version>
</dependency>
```

Note

Before the release of the `HttpDestination` library to the Maven central repository, SAP BTP provided a library based on Apache `HttpClient` 4.1.3, which had several limitations.

Consume HttpDestination

Features

- The recent versions of Apache `HttpClient` provide some new features (like async processing), better practices, etc.
- The SAP-specific library includes these main differences compared to the older versions:
  - Changed configuration options for `HttpClient` parameters
  - Usage of ClosableHttpClient
  - Several deprecated methods and classes

For more information, see release notes .

PoolingConnectionManager Properties

With the new `HttpClient`, you cannot change the configuration of the pooling connection manager once it is created.

Therefore, if a client wants to configure the `maxTotalConnections` and `defaultMaxConnectionsPerRoute` properties of the connection manager, you must configure them as properties in the destination itself.

The Apache API defines the following methods for the `PoolingConnectionManager`:

Sample Code

```
setMaxTotal(int max); //Set the maximum number of total open connections. 20
by default.

setDefaultMaxPerRoute(int max); // Set the maximum number of concurrent
connections per route, which is 2 by default.
```

You can change the values by defining them in the *additional properties* of the HTTP destination:

- `MAX_TOTAL_CONNECTIONS` (`maxTotalConnections`)
- `DEFAULT_MAX_CONNECTIONS_PER_ROUTE` (`defaultMaxConnectionsPerRoute`)

See also Create HTTP Destinations [page 80].

HttpClient Class

As the `HttpClient` class implements `java.io.Closable`, a new class is introduced on SAP BTP side: `com.sap.core.connectivity.api.http.HttpDestinationClient`.

This class abstracts `org.apache.http.client.HttpClient`, and is the one that you should use.

1.4.4.1.2.2 HTTP Proxy for On-Premise Connectivity

The Connectivity service provides a standard HTTP Proxy for on-premise connectivity that is accessible by any application.

Context [page 152]

Multitenancy Support [page 152]

Using the Proxy on Multi-Tenant VMs [page 153]

Using the Proxy on Single-Tenant VMs [page 153]

Context

Proxy host and port are available as the environment variables `HC_OP_HTTP_PROXY_HOST` and `HC_OP_HTTP_PROXY_PORT`.

> **i** Note
>
> - The HTTP Proxy provides a more flexible way to use on-premise connectivity via standard HTTP clients. It is not suitable for other protocols, such as `RFC` or `Mail`. HTTPS requests will not work as well.
> - The previous alternative, that is, using on-premise connectivity via the existing HTTP Destination API, is still supported. For more information, see HttpDestination Library [page 147].

> **⚠** Caution
>
> There is a limit of **8192** bytes for the size of the HTTP lines (for example, request line or header) that you send via the Connectivity service. If this limit is exceeded, you receive an HTTP error of type 4xx. This issue is usually caused by the size of the *path* + *query* string of the request.

Multitenancy Support

By default, all applications are started in multitenant mode. Such applications are responsible to propagate consumer subaccounts to the HTTP Proxy, using the header **`SAP-Connectivity-ConsumerAccount`**. This header is mandatory during the first request of each HTTP connection. HTTP connections are associated with one consumer subaccount and cannot be used with another subaccount. If the **`SAP-Connectivity-ConsumerAccount`** header is sent after the first request, and its value is different from the value in the first request, the Proxy will return HTTP response code `400`.

Starting with SAP HANA Cloud Connector 2.9.0, it is possible to connect multiple Cloud Connectors to a subaccount as long as their location ID is different. Using the header `SAP-Connectivity-SCC-Location_ID` it is possible to specify the Cloud Connector over which the connection is opened. If this header is not specified, the connection will be opened to the Cloud Connector that is connected without any location ID. This is also the case for all Cloud Connector versions prior to 2.9.0.

If an application virtual machine (VM) is started for one consumer subaccount, this subaccount is known by the HTTP Proxy and the application may not send the `SAP-Connectivity-ConsumerAccount` header.

Using the Proxy on Multi-Tenant VMs

On multitenant VMs, applications are responsible to propagate consumer subaccount via `SAP-Connectivity-ConsumerAccount` header. The following example shows how this can be performed.

```
// TenantContex instance injection. It is used to get the consumer subaccount
name.
@Resource
public TenantContext  tenantContext;
...

String proxyHost = System.getenv("HC_OP_HTTP_PROXY_HOST");
int proxyPort = Integer.parseInt(System.getenv("HC_OP_HTTP_PROXY_PORT"));

// set up the on-premise HTTP Proxy
HttpClient httpClient = new DefaultHttpClient();
httpClient.getParams().setParameter(ConnRoutePNames.DEFAULT_PROXY, new
HttpHost(proxyHost, proxyPort));

// insert the necessary headers in the request
HttpGet request = new HttpGet("http://virtualhost:1234");
request.addHeader("SAP-Connectivity-ConsumerAccount",
tenantContext.getTenant().getAccount().getId());

// execute the request
HttpResponse response = httpClient.execute(request);
```

Using the Proxy on Single-Tenant VMs

On single-tenant VMs, the consumer subaccount is known and subaccount propagation via header is not needed. The following example demonstrates this case.

```
String proxyHost = System.getenv("HC_OP_HTTP_PROXY_HOST");
int proxyPort = Integer.parseInt(System.getenv("HC_OP_HTTP_PROXY_PORT"));

// create HTTP client and insert the necessary headers in the request
HttpClient httpClient = new DefaultHttpClient();
httpClient.getParams().setParameter(ConnRoutePNames.DEFAULT_PROXY, new
HttpHost(proxyHost, proxyPort));
HttpGet request = new HttpGet("http://virtualhost:1234");

// execute the request
HttpResponse response = httpClient.execute(request);
```

Related Information

Connectivity and Destination APIs [page 135]

Principal Propagation Using HTTP Proxy [page 154]

1.4.4.1.2.2.1 Principal Propagation Using HTTP Proxy

Context

The HTTP Proxy can forward the identity of an on-demand user to the Cloud Connector, and from there – to the back-end of the relevant on-premise system. In this way, on-demand users will no longer need to provide their identity every time they make connections to on-premise systems via one and the same Cloud Connector. To propagate the logged-in user, an application must use the `AuthenticationHeaderProvider` API to generate a header, which then embeds in the HTTP request to the on-premise system.

Restrictions

- IDPs used by applications protected by SAML2 have to be denoted as trustworthy for the Cloud Connector.
- Non-SAML2 protected applications have to be denoted themselves as trustworthy for the Cloud Connector.

Example

```
String proxyHost = System.getenv("HC_OP_HTTP_PROXY_HOST");
int proxyPort = Integer.parseInt(System.getenv("HC_OP_HTTP_PROXY_PORT"));
String account = System.getenv("HC_ACCOUNT");

// setup the on-premise HTTP proxy
HttpClient httpClient = new DefaultHttpClient();
httpClient.getParams().setParameter(ConnRoutePNames.DEFAULT_PROXY, new
HttpHost(proxyHost, proxyPort));

// look up the connectivity authentication header provider resource called
"authHeaderProvider" (must be defined in web.xml)
Context ctx = new InitialContext();
AuthenticationHeaderProvider authHeaderProvider = (AuthenticationHeaderProvider)
ctx.lookup("java:comp/env/authHeaderProvider");
// get header for principal propagation
AuthenticationHeader principalPropagationHeader =
authHeaderProvider.getPrincipalPropagationHeader();

//insert the necessary headers in the request
HttpGet request = new HttpGet("http://virtualhost:1234");
request.addHeader(principalPropagationHeader.getName(),
principalPropagationHeader.getValue());
request.addHeader("SAP-Connectivity-ConsumerAccount", account);
```

```
// execute the request
HttpResponse response = httpClient.execute(request);
```

> **i** Note
>
> You can also apply dependency injection by using the @Resource annotation.

Related Information

AuthenticationHeaderProvider API [page 141]
HTTP Proxy for On-Premise Connectivity [page 152]

1.4.4.1.2.3 Use Cases

Overview

The Connectivity service enables access to remote services running either on the Internet or in an on-premise network.

Use Cases

The examples in this section show how you can make connections to Internet services and on-premise networks:

Consume Internet Services (Java Web or Java EE 6 Web Profile) [page 155]

Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170]

1.4.4.1.2.3.1 Consume Internet Services (Java Web or Java EE 6 Web Profile)

Context

This example demonstrates consumption of Internet services using Apache HTTP Client . The example also shows how a connectivity-enabled Web application can be deployed on a local server and on the cloud.

The servlet code, the `web.xml` content, and the destination file (**`outbound-internet-destination`**) used in this example are mapped to the connectivity sample project located in `<SDK_location>/samples/connectivity`. You can directly import this sample in your Eclipse IDE. For more information, see .

Go through the relevant steps:

1. Create a Dynamic Web Project [page 196]
2. Create a Sample Servlet [page 197]
3. Test the Connectivity-Enabled Web Application Locally [page 160]
4. Deploy the Connectivity-Enabled Web Application on the Cloud [page 161]

Prerequisites

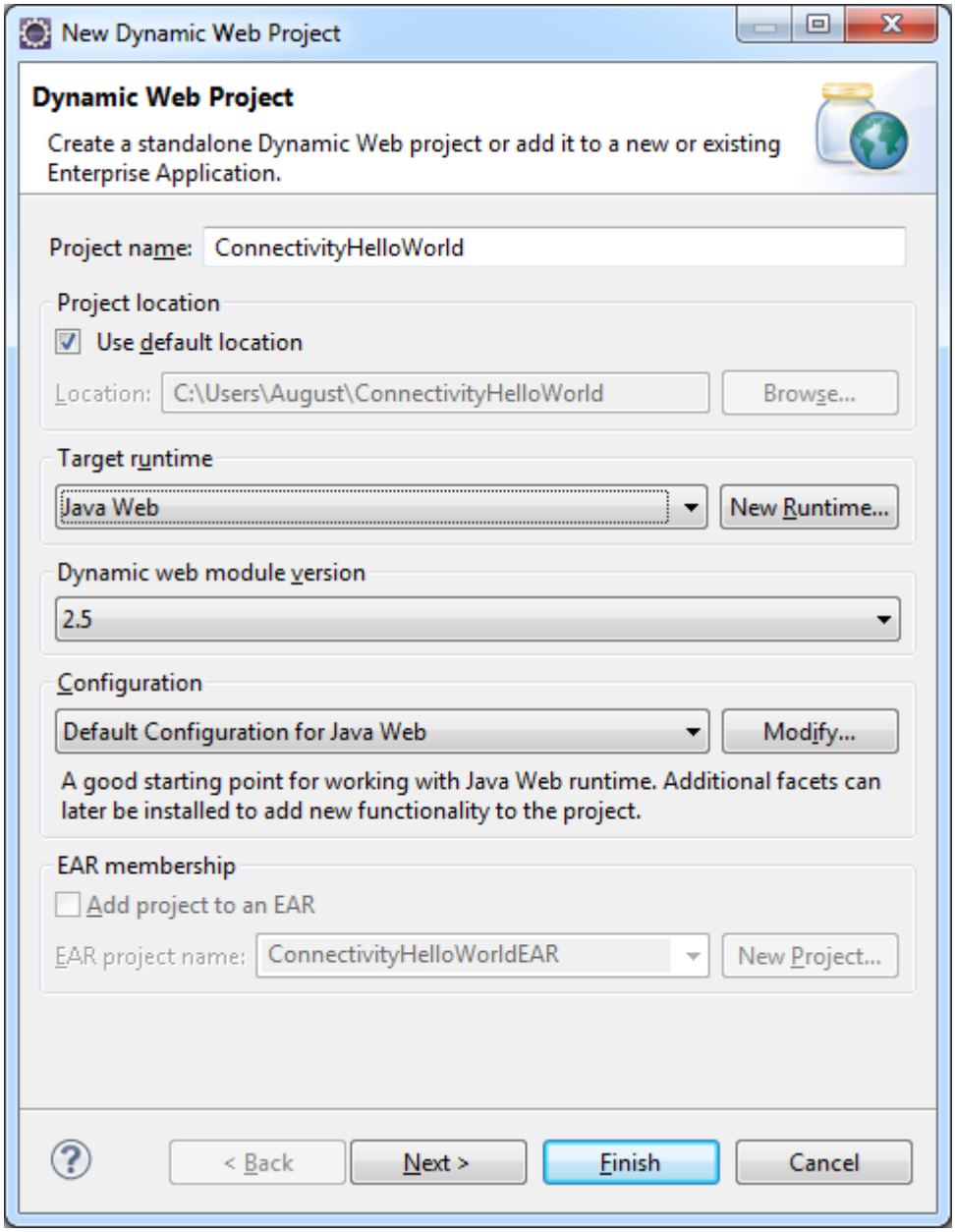
You have downloaded and set up your Eclipse IDE, SAP BTP Tools for Java, and SDK.

For more information, see Setting Up the Development Environment [page 845].

> **i** Note
>
> You need to install **`SDK for Java Web`** or **`SDK for Java EE 6 Web Profile`**.

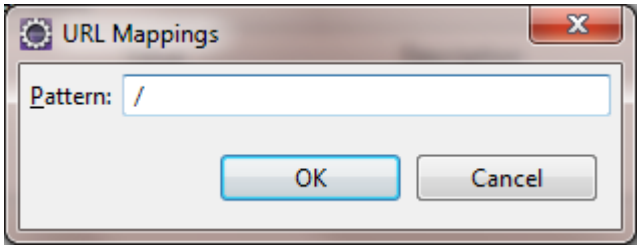
1. Create a Dynamic Web Project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. From the Eclipse main menu, choose *File* > *New* > *Dynamic Web Project* .
3. In the *Project name* field, enter **`ConnectivityHelloWorld`** .
4. In the *Target Runtime* pane, select the runtime you want to use to deploy the application. In this example, we choose **`Java Web`**.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to finalize the creation of your project.



2. Create a Sample Servlet

1. From the *ConnectivityHelloWorld* context menu, choose *New* > *Servlet*.
2. Enter `hello` as the Java package and `ConnectivityServlet` as the *Class name* and choose *Next*.
3. In the *URL mappings* field, select */ConnectivityServlet* and choose *Edit*.
4. In the *Pattern* field, replace the current value with just "`/`". In this way, the servlet will be mapped as a welcome page for the application.



5. Choose *Finish* so that the `ConnectivityServlet.java` servlet is created and opened in the Java editor.
6. Go to *ConnectivityHelloWorld* > *WebContent* > *WEB-INF* and open the `web.xml` file.
7. Choose the *Source* tab page.
8. Add the following code block to the `<web-app>` element:

```
<resource-ref>
      <res-ref-name>outbound-internet-destination</res-ref-name>
      <res-type>com.sap.core.connectivity.api.http.HttpDestination</res-type>
</resource-ref>
```

> **i Note**
>
> The value of the `<res-ref-name>` element in the `web.xml` file should match the name of the destination that you want to be retrieved at runtime. In this case, the destination name is **`outbound-internet-destination`**.

9. Replace the entire servlet class with the following one to make use of the destination API. The destination API is visible by default for cloud applications and must not be added explicitly to the application class path.

```
package com.sap.cloud.sample.connectivity;
import java.io.IOException;
import java.io.InputStream;
import static java.net.HttpURLConnection.HTTP_OK;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.sap.core.connectivity.api.DestinationFactory;
import com.sap.core.connectivity.api.http.HttpDestination;
/**
 * Servlet class making HTTP calls to specified HTTP destinations.
 * Destinations are used in the following exemplary connectivity
scenarios:<br>
 * - Connecting to an outbound Internet resource using HTTP destinations<br>
 * - Connecting to an on-premise backend using on-premise HTTP
destinations,<br>
 *   where the destinations could have no authentication or basic
authentication.<br>
 *
 * * NOTE: The Connectivity
                                    service API is located under
 * <code>com.sap.core.connectivity.api</code>. The old API under
```

```
 * <code>com.sap.core.connectivity.httpdestination.api</code> has been
deprecated.
 */
public class ConnectivityServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final int COPY_CONTENT_BUFFER_SIZE = 1024;
    private static final Logger LOGGER =
LoggerFactory.getLogger(ConnectivityServlet.class);
    /** {@inheritDoc} */
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpClient httpClient = null;
        String destinationName = request.getParameter("destname");
        try {
            // Get HTTP destination
            Context ctx = new InitialContext();
            HttpDestination destination = null;
            if (destinationName != null) {
                DestinationFactory destinationFactory = (DestinationFactory)
ctx.lookup(DestinationFactory.JNDI_NAME);
                destination = (HttpDestination)
destinationFactory.getDestination(destinationName);
            } else {
                // The default request to the Servlet will use outbound-
internet-destination
                destinationName = "outbound-internet-destination";
                destination = (HttpDestination) ctx.lookup("java:comp/env/"
+  destinationName);
            }
            // Create HTTP client
            httpClient = destination.createHttpClient();
            // Execute HTTP request
            HttpGet httpGet = new HttpGet();
            HttpResponse httpResponse = httpClient.execute(httpGet);
            // Check response status code
            int statusCode = httpResponse.getStatusLine().getStatusCode();
            if (statusCode != HTTP_OK) {
                throw new ServletException("Expected response status code is
200 but it is " + statusCode + " .");
            }
            // Copy content from the incoming response to the outgoing
response
            HttpEntity entity = httpResponse.getEntity();
            if (entity != null) {
                InputStream instream = entity.getContent();
                try {
                    byte[] buffer = new byte[COPY_CONTENT_BUFFER_SIZE];
                    int len;
                    while ((len = instream.read(buffer)) != -1) {
                        response.getOutputStream().write(buffer, 0, len);
                    }
                } catch (IOException e) {
                    // In case of an IOException the connection will be
released
                    // back to the connection manager automatically
                    throw e;
                } catch (RuntimeException e) {
                    // In case of an unexpected exception you may want to
abort
                    // the HTTP request in order to shut down the underlying
                    // connection immediately.
                    httpGet.abort();
                    throw e;
                } finally {
                    // Closing the input stream will trigger connection
release
                    try {
```

```
                        instream.close();
                    } catch (Exception e) {
                    // Ignore
                    }
                }
            }
        } catch (NamingException e) {
            // Lookup of destination failed
            String errorMessage = "Lookup of destination failed with reason: "
                    + e.getMessage()
                    + ". See "
                    + "logs for details. Hint: Make sure to have the
destination "
                    + destinationName + " configured.";
            LOGGER.error("Lookup of destination failed", e);
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                    errorMessage);
        } catch (Exception e) {
            // Connectivity operation failed
            String errorMessage = "Connectivity operation failed with reason:
"
                    + e.getMessage()
                    + ". See "
                    + "logs for details. Hint: Make sure to have an HTTP
proxy configured in your "
                    + "local Eclipse environment in case your environment
uses "
                    + "an HTTP proxy for the outbound Internet "
                    + "communication.";
            LOGGER.error("Connectivity operation failed", e);
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                    errorMessage);
        } finally {
            // When HttpClient instance is no longer needed, shut down the
connection manager to ensure immediate
            // deallocation of all system resources
            if (httpClient != null) {
                httpClient.getConnectionManager().shutdown();
            }
        }
    }
}
```

> **i Note**
>
> The given servlet can run with different destination scenarios, for which user should specify the destination name as a requested parameter in the calling URL. In this case, the destination name should be `<applicationURL>/?destname=outbound-internet-destination`. Nevertheless, your servlet can still run even without specifying the destination name for this outbound scenario.

10. Save the Java editor and make sure the project compiles without errors.

3. Test the Connectivity-Enabled Web Application Locally

> **⚠ Caution**
>
> - If you use SDK for Java Web, we only recommend that you create a destination before deploying the application.
> - If you use SDK for Java EE 6 Web Profile, you must create a destination before deploying the application.

1. In the context menu of the *Servers* view, choose *New* > *Server*.
2. Expand the *SAP* node, select *Java Web Server* and choose *Finish*.
3. A new server `Java Web Server [Stopped, Synchronized]` appears on the *Servers* tab page. Also, a *Servers* folder is created and appears in the navigation tree of the IDE. It contains configurable folders and files you can use, for example, to change your HTTP or JMX ports.
4. If you work behind a proxy server, you need to configure your proxy setting as follows:
   - In the *Servers* view, double-click the added server to open the editor.
   - Click the *Open Launch Configuration* link.
   - Choose the *(x)=Arguments* tab page.
   - In the *VM Arguments* box, add the following row:

     ```
     -Dhttp.proxyHost=<your_proxy_host> -Dhttp.proxyPort=<your_proxy_port> -
     Dhttps.proxyHost=<your_proxy_host> -Dhttps.proxyPort=<your_proxy_port>
     ```

   - Choose *OK*.
5. Go to the *Connectivity* tab page of your local server. Create a destination with the name **`outbound-internet-destination`**, and configure it so it can be consumed by the application at runtime. For more information, see Configure Destinations from the Eclipse IDE [page 64].
   For the sample destination to work properly, the following properties need to be configured:

   ```
   Name=outbound-internet-destination
   Type=HTTP
   URL=http://sap.com/index.html
   Authentication=NoAuthentication
   ```

6. From the *ConnectivityServlet.java* editor's context menu, choose *Run As* > *Run on Server*.
7. Make sure that the *Choose an existing server* option is selected and choose *Java Web Server*.
8. Choose *Finish*.
   The server is now started, displayed as `Java Web Server [Started, Synchronized]` in the *Servers* view.

Result:

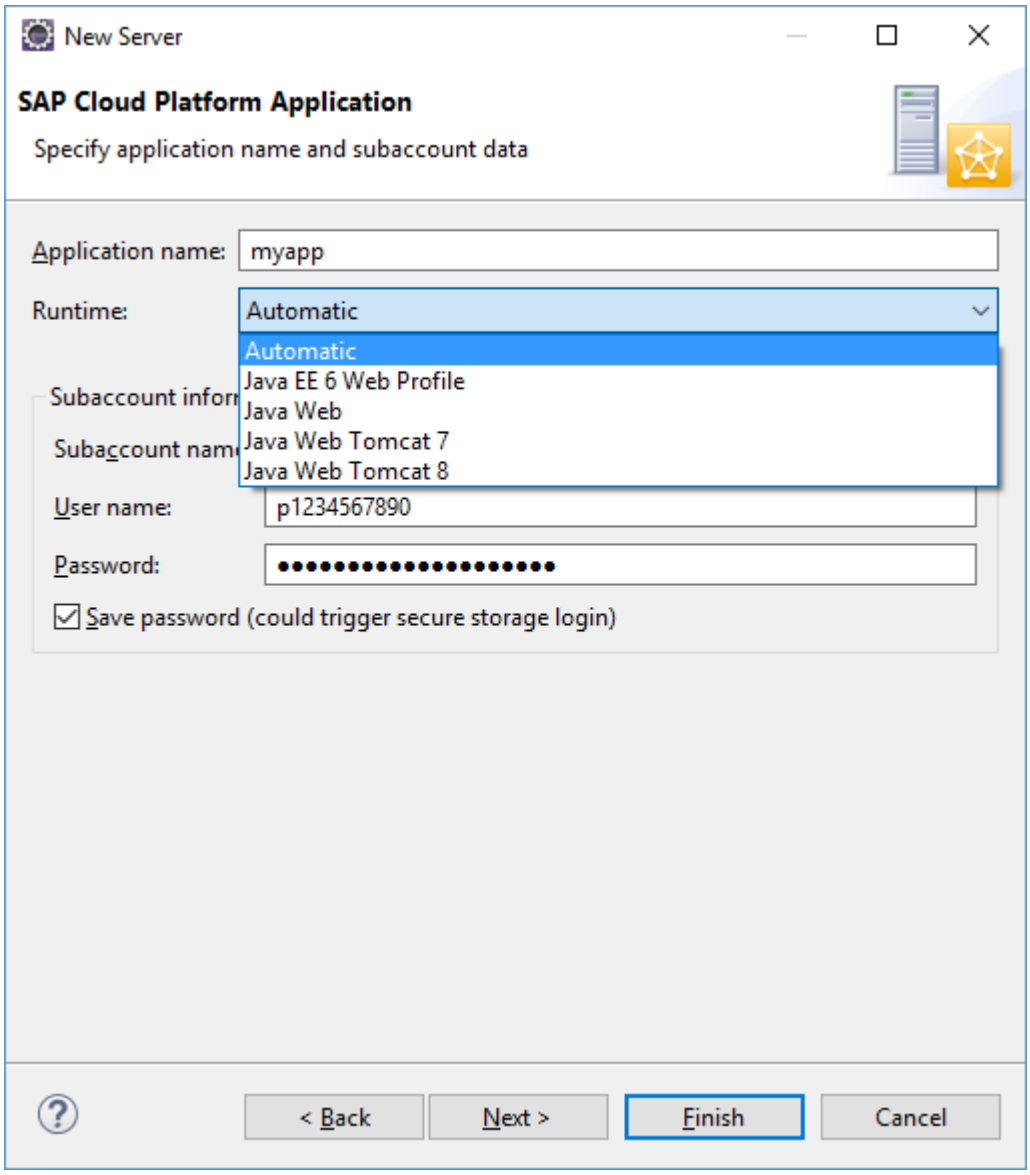
The internal Web browser opens with the expected output of the connectivity-enabled Web application.

4. Deploy the Connectivity-Enabled Web Application on the Cloud

1. In the context menu of the *Servers* view, choose *New* > *Server*.
2. Choose SAP BTP as the type of server you want to create and choose *Next*.
3. For *Server's host name*, specify the region host depending on your subaccount type. For more information, see Regions and Hosts Available for the Neo Environment [page 17].
4. Choose *Next*.
5. On the *New Server* wizard page, enter your application and subaccount name. Note that only lowercase Latin letters and digits are allowed.

   > **i Note**
   >
   > The application name should be unique enough to allow your deployed application to be easily identified in SAP BTP cockpit.

6. Enter your subaccount name, e-mail or user name, and password.



7. Choose *Finish*.
8. A new server `<application>.<subaccount> [Stopped]>` appears in the *Servers* view.
9. Go to the *Connectivity* tab page of the server, create a destination with the name **`outbound-internet-destination`**, and configure it using the following properties:

```
Name=outbound-internet-destination
Type=HTTP
URL=http://sap.com/index.html
Authentication=NoAuthentication
ProxyType=Internet
```

10. From the *ConnectivityServlet.java* editor's context menu, choose *Run As* > *Run on Server*.
11. Make sure that the *Choose an existing server* option is selected and choose *<Server_host_name>* > *<Server_name>*.
12. Choose *Finish*.

Result:

The internal Web browser opens with the URL pointing to SAP BTP and displaying the expected output of the connectivity-enabled Web application.

Next Step

You can monitor the state and logs of your Web application deployed on SAP BTP.

For more information, see .

1.4.4.1.2.3.2 Consume Internet Services (Java Web Tomcat 7)

Context

This example demonstrates consumption of Internet services using `HttpURLConnection`. The example also shows how a connectivity-enabled Web application can be deployed on a local server and on the cloud.

The servlet code, the `web.xml` content, and the destination file (**`outbound-internet-destination`**) used in this example are mapped to the connectivity sample project located in `<SDK_location>/samples/connectivity`. You can directly import this sample in your Eclipse IDE. For more information, see .

Go through the relevant steps:

1. Create a Dynamic Web Project [page 196]
2. Create a Sample Servlet [page 197]
3. Test the Connectivity-Enabled Web Application Locally [page 168]
4. Deploy the Connectivity-Enabled Web Application on the Cloud [page 168]

Prerequisites

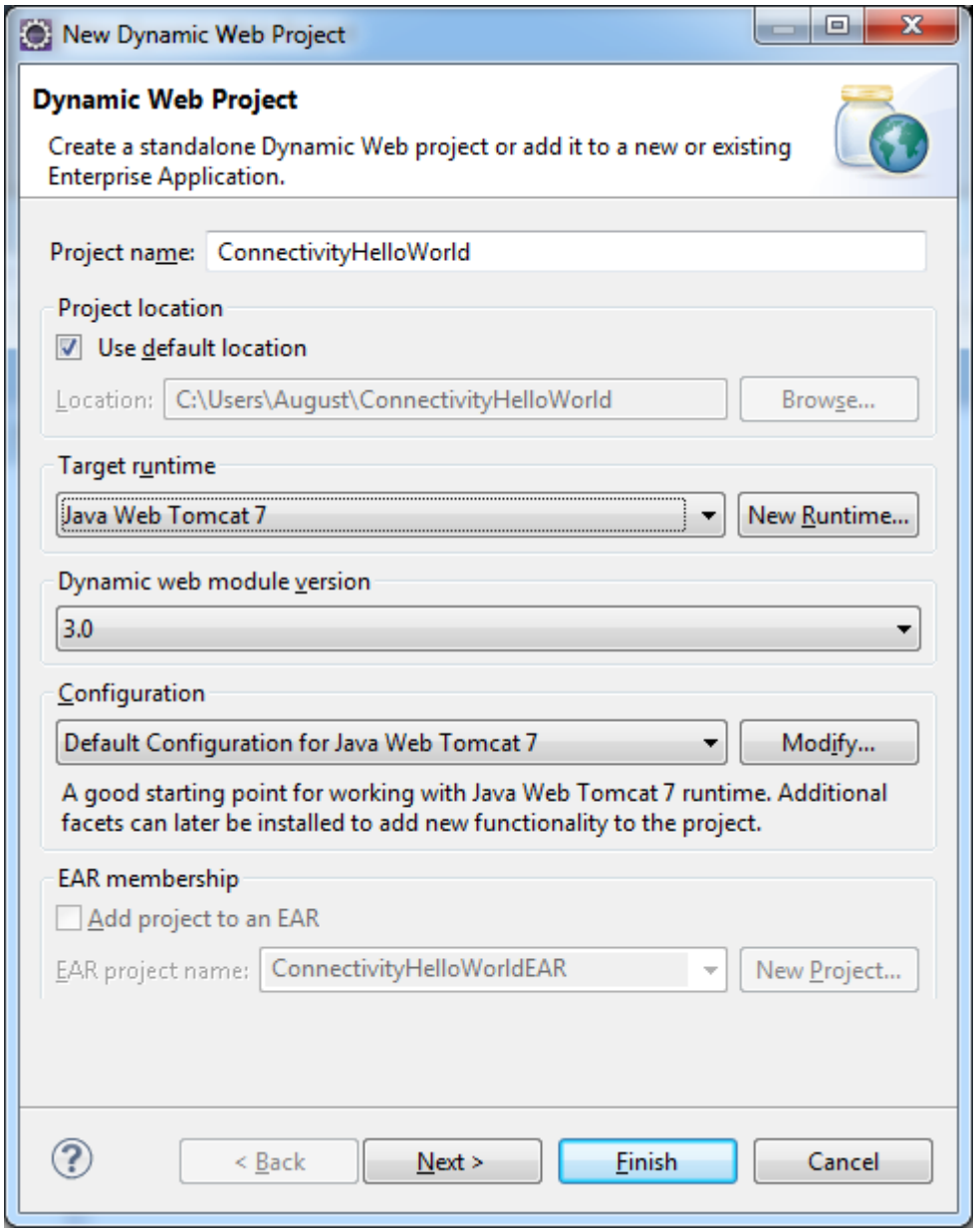
You have downloaded and set up your Eclipse IDE, SAP BTP Tools for Java, and SDK.

For more information, see Setting Up the Development Environment [page 845].

> **i** Note
>
> You need to install **`SDK for Java Web Tomcat 7`**.

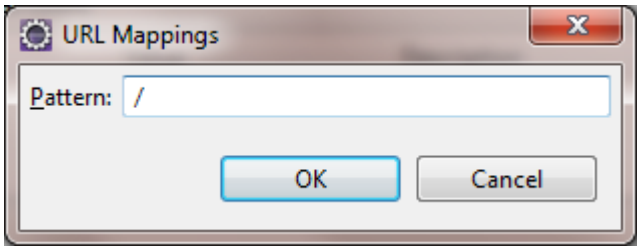
1. Create a Dynamic Web Project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. From the Eclipse main menu, choose *File* > *New* > *Dynamic Web Project*.
3. In the *Project name* field, enter `ConnectivityHelloWorld` .
4. In the *Target Runtime* pane, select `Java Web Tomcat 7` as the runtime you want to use to deploy the application.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to finalize the creation of your project.



2. Create a Sample Servlet

1. From the *ConnectivityHelloWorld* context menu, choose *New* > *Servlet*.
2. Enter **`hello`** as the Java package and **`ConnectivityServlet`** as the *Class name* and choose *Next*.
3. In the *URL mappings* field, select */ConnectivityServlet* and choose *Edit*.
4. In the *Pattern* field, replace the current value with just "**`/`**". In this way, the servlet will be mapped as a welcome page for the application.



5. Choose *Finish* so that the `ConnectivityServlet.java` servlet is created and opened in the Java editor.
6. Go to *ConnectivityHelloWorld* > *WebContent* > *WEB-INF* and open the `web.xml` file.
7. Choose the *Source* tab page.
8. To consume connectivity configuration using JNDI, you need to define the `ConnectivityConfiguration` API as a resource in the `web.xml` file. Below is an example of a `ConnectivityConfiguration` resource, named **`connectivityConfiguration`**.

```
<resource-ref>
    <res-ref-name>connectivityConfiguration</res-ref-name>
    <res-
type>com.sap.core.connectivity.api.configuration.ConnectivityConfiguration</
res-type>
</resource-ref>
```

9. Replace the entire servlet class with the following one to make use of the destination API. The destination API is visible by default for cloud applications and must not be added explicitly to the application class path.

```
package com.sap.cloud.sample.connectivity;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.InetSocketAddress;
import java.net.Proxy;
import java.net.URL;

import javax.annotation.Resource;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.sap.cloud.account.TenantContext;
import com.sap.core.connectivity.api.configuration.ConnectivityConfiguration;
import com.sap.core.connectivity.api.configuration.DestinationConfiguration;
```

```
/**
 * Servlet class making http calls to specified http destinations.
 * Destinations are used in the following example connectivity scenarios:<br>
 * - Connecting to an outbound Internet resource using HTTP destinations<br>
 * - Connecting to an on-premise backend using on premise HTTP
destinations,<br>
 *   where the destinations have no authentication.<br>
 */
public class ConnectivityServlet extends HttpServlet {
    @Resource
    private TenantContext  tenantContext;

    private static final long serialVersionUID = 1L;
    private static final int COPY_CONTENT_BUFFER_SIZE = 1024;
    private static final Logger LOGGER =
LoggerFactory.getLogger(ConnectivityServlet.class);

    private static final String ON_PREMISE_PROXY = "OnPremise";

    /** {@inheritDoc} */
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpURLConnection urlConnection = null;
        String destinationName = request.getParameter("destname");

        // The default request to the Servlet will use outbound-internet-
destination
        if (destinationName == null) {
            destinationName = "outbound-internet-destination";
        }

        try {
            // Look up the connectivity configuration API
            Context ctx = new InitialContext();
            ConnectivityConfiguration configuration =
(ConnectivityConfiguration) ctx.lookup("java:comp/env/
connectivityConfiguration");

            // Get destination configuration for "destinationName"
            DestinationConfiguration destConfiguration =
configuration.getConfiguration(destinationName);
            if (destConfiguration == null) {

response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                        String.format("Destination %s is not found. Hint:
Make sure to have the destination configured.", destinationName));
                return;
            }

            // Get the destination URL
            String value = destConfiguration.getProperty("URL");
            URL url = new URL(value);

            String proxyType = destConfiguration.getProperty("ProxyType");
            Proxy proxy = getProxy(proxyType);

            urlConnection = (HttpURLConnection) url.openConnection(proxy);

            // Insert the required header in the request for on-premise
destinations
            injectHeader(urlConnection, proxyType);

            // Copy content from the incoming response to the outgoing
response
            InputStream instream = urlConnection.getInputStream();
            OutputStream outstream = response.getOutputStream();
```

```
            copyStream(instream, outstream);
        } catch (Exception e) {
            // Connectivity operation failed
            String errorMessage = "Connectivity operation failed with reason:
"
                    + e.getMessage()
                    + ". See "
                    + "logs for details. Hint: Make sure to have an HTTP
proxy configured in your "
                    + "local environment in case your environment uses "
                    + "an HTTP proxy for the outbound Internet "
                    + "communication.";
            LOGGER.error("Connectivity operation failed", e);
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                    errorMessage);
        }
    }

    private Proxy getProxy(String proxyType) {
        String proxyHost = null;
        int proxyPort;

        if (ON_PREMISE_PROXY.equals(proxyType)) {
            // Get proxy for on-premise destinations
            proxyHost = System.getenv("HC_OP_HTTP_PROXY_HOST");
            proxyPort =
Integer.parseInt(System.getenv("HC_OP_HTTP_PROXY_PORT"));
        } else {
            // Get proxy for internet destinations
            proxyHost = System.getProperty("http.proxyHost");
            proxyPort =
Integer.parseInt(System.getProperty("http.proxyPort"));
        }
        return new Proxy(Proxy.Type.HTTP, new InetSocketAddress(proxyHost,
proxyPort));
    }

    private void injectHeader(HttpURLConnection urlConnection, String
proxyType) {
        if (ON_PREMISE_PROXY.equals(proxyType)) {
            // Insert header for on-premise connectivity with the consumer
subaccount name
            urlConnection.setRequestProperty("SAP-Connectivity-
ConsumerAccount", tenantContext.getAccountName());
        }
    }

    private void copyStream(InputStream inStream, OutputStream outStream)
throws IOException {
        byte[] buffer = new byte[COPY_CONTENT_BUFFER_SIZE];
        int len;
        while ((len = inStream.read(buffer)) != -1) {
            outStream.write(buffer, 0, len);
        }
    }
}
```

> ### Note
>
> The given servlet can run with different destination scenarios, for which user should specify the destination name as a requested parameter in the calling URL. In this case, the destination name should be `<applicationURL>/?destname=outbound-internet-destination`. Nevertheless, your servlet can still run even without specifying the destination name for this outbound scenario.

10. Save the Java editor and make sure the project compiles without errors.

3. Test the Connectivity-Enabled Web Application Locally

> **i Note**
>
> We recommend but not obligate that you create a destination before deploying the application.

1. In the context menu of the *Servers* view, choose *New* > *Server* .
2. Expand the *SAP* node, select *Java Web Tomcat 7 Server* and choose *Finish*.
3. A new server `Java Web Tomcat 7 Server [Stopped, Synchronized]` appears on the *Servers* tab page.
   Also, a *Servers* folder is created and appears in the navigation tree of the IDE. It contains configurable folders and files you can use, for example, to change your HTTP or JMX ports.
4. If you work behind a proxy server, you need to configure your proxy setting as follows:
   - In the *Servers* view, double-click the added server to open the editor.
   - Click the *Open Launch Configuration* link.
   - Choose the *(x)=Arguments* tab page.
   - In the *VM Arguments* box, add the following row:

     ```
     -Dhttp.proxyHost=<your_proxy_host> -Dhttp.proxyPort=<your_proxy_port> -
     Dhttps.proxyHost=<your_proxy_host> -Dhttps.proxyPort=<your_proxy_port>
     ```

   - Choose *OK*.
5. Go to the *Connectivity* tab page of your local server, create a destination with the name **`outbound-internet-destination`**, and configure it so it can be consumed by the application at runtime. For more information, see Configure Destinations from the Eclipse IDE [page 64].
   For the sample destination to work properly, the following properties need to be configured:

   ```
   Name=outbound-internet-destination
   Type=HTTP
   URL=http://sap.com/index.html
   Authentication=NoAuthentication
   ```

6. From the *ConnectivityServlet.java* editor's context menu, choose *Run As* > *Run on Server* .
7. Make sure that the *Choose an existing server* option is selected and choose *Java Web Tomcat 7 Server*.
8. Choose *Finish*.
   The server is now started, displayed as `Java Web Tomcat 7 Server [Started, Synchronized]` in the *Servers* view.

Result:

The internal Web browser opens with the expected output of the connectivity-enabled Web application.

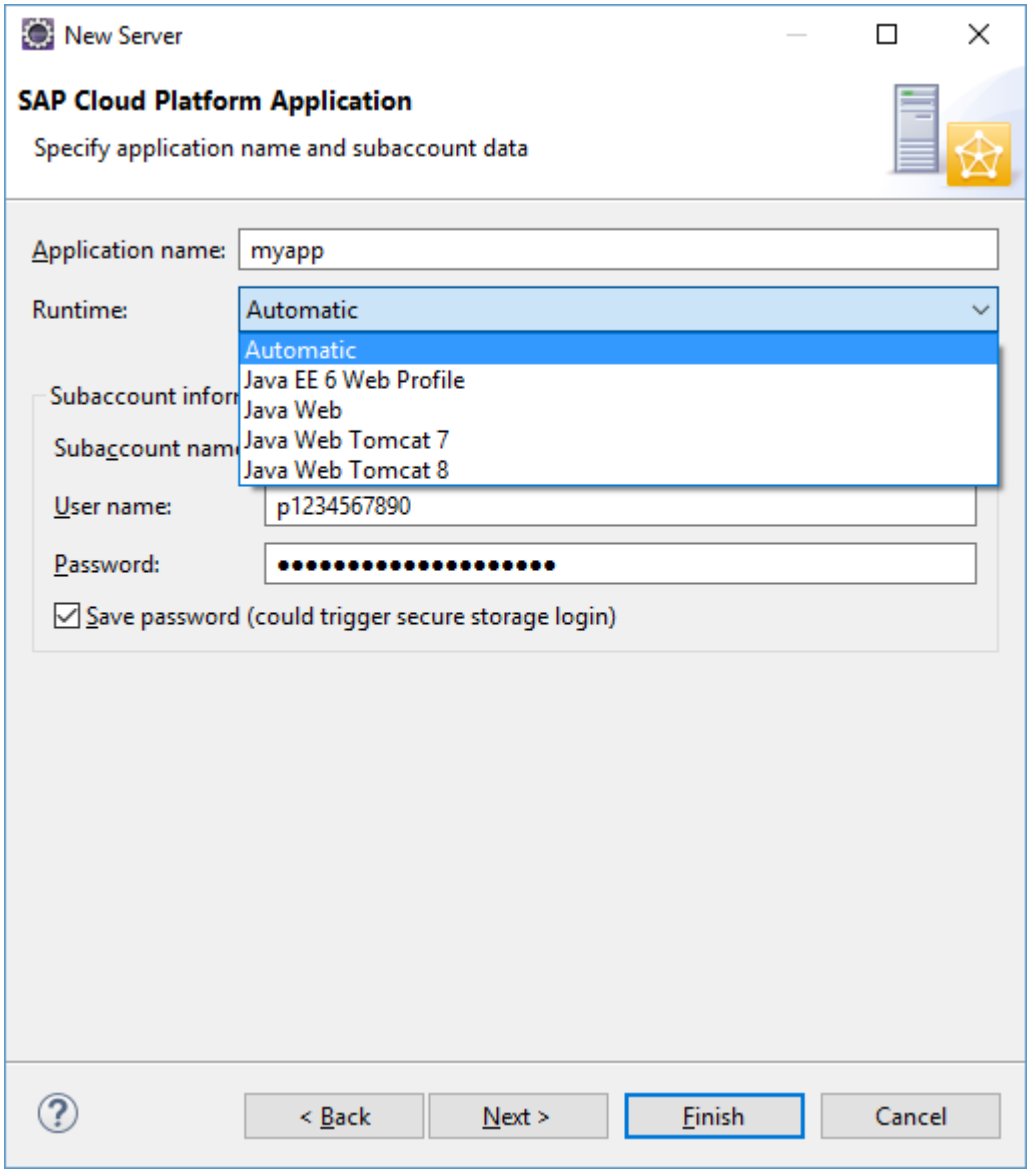
4. Deploy the Connectivity-Enabled Web Application on the Cloud

1. In the context menu of the *Servers* view, choose *New* > *Server* .
2. Choose *SAP Cloud Platform* as the type of server you want to create and choose *Next*.
3. For *Server's host name*, specify the region host depending on your subaccount type. For more information, see Regions and Hosts Available for the Neo Environment [page 17].

4. Choose *Next*.
5. On the *New Server* wizard page, enter your application and subaccount name. Note that only lowercase Latin letters and digits are allowed.

> **i Note**
>
> The application name should be unique enough to allow your deployed application to be easily identified in SAP BTP cockpit.

6. Enter your subaccount name, e-mail or user name, and password.



7. Choose *Finish*.
8. A new server `<application>.<subaccount> [Stopped]` appears in the *Servers* view.
9. Go to the *Connectivity* tab page of the server. Create a destination with the name **`outbound-internet-destination`**, and configure it using the following properties:

```
Name=outbound-internet-destination
Type=HTTP
```

```
URL=http://sap.com/index.html
Authentication=NoAuthentication
ProxyType=Internet
```

10. From the *ConnectivityServlet.java* editor's context menu, choose *Run As* > *Run on Server*.
11. Make sure that the *Choose an existing server* option is selected and choose *<Server_host_name>* > *<Server_name>*.
12. Choose *Finish*.

Result:

The internal Web browser opens with the URL pointing to SAP BTP and displaying the expected output of the connectivity-enabled Web application.

Next Step

You can monitor the state and logs of your Web application deployed on SAP BTP.

For more information, see .

1.4.4.1.2.3.3 Consume Backend Systems (Java Web or Java EE 6 Web Profile)

Context

This example demonstrates how a sample Web application consumes a backend system via HTTP(S) by using the Connectivity service. For simplicity, instead of using a real backend system, we use a second sample Web application containing `BackendServlet`. It mimics the backend system and can be called via HTTP(S).

The servlet code, the `web.xml` content, and the destination files (**`backend-no-auth-destination`** and **`backend-basic-auth-destination`**) used in this example are mapped to the connectivity sample project located in `<SDK_location>/samples/connectivity`. You can directly import this sample in your Eclipse IDE. For more information, see .

The example includes the following sections:

1. Set Up Application as a Backend System [page 171]
2. Create a Dynamic Web Project [page 196]
3. Create a Sample Servlet [page 174]
4. Deploy the Application [page 199]
5. Configure the Destination in the Cloud [page 179]

Connectivity User Roles

In the on-demand to on-premise connectivity end-to-end scenario, different user roles are involved. The particular steps for the relevant roles are described below:

- *IT Administrator* - Sets up and configures the Cloud Connector. Scenario steps:
  1. Downloads the Cloud Connector from https://tools.hana.ondemand.com/#cloud
  2. Installs the connector.
  3. Establishes an SSL tunnel from the connector to an SAP BTP subaccount.
  4. Configures the exposed backend systems and resources.
- *Application Developer* - Develops Web applications using destinations. Scenario steps:
  1. Installs the Eclipse IDE, SAP BTP Tools for Java, and SDK.
  2. Develops a Java EE application using the destination API.
  3. Configures connectivity destinations as resources in the `web.xml` file.
  4. Configures connectivity destinations via the SAP BTP server adapter in Eclipse IDE.
  5. Deploys the Java EE application locally and on the cloud.
- *Subaccount Operator* - Deploys Web applications, configures their destinations, and conducts tests. Scenario steps:
  1. Obtains a ready Java EE application WAR file.
  2. Deploys the Java EE application to an SAP BTP subaccount.
  3. Uploads the connectivity destination configuration via the console client.
  4. Tests the Java EE application on a local server and deploys it again to an SAP BTP subaccount.

For more information, see Cloud Connector [page 232].

Prerequisites

- You have downloaded and configured the Cloud Connector. For more information, see Cloud Connector [page 232].
- You have downloaded and set up your Eclipse IDE, SAP BTP Tools for Java, and SDK.
  For more information, see Setting Up the Development Environment [page 845].

  > **i Note**
  >
  > You need to install **`SDK for Java Web`** or **`SDK for Java EE 6 Web Profile`**.

1. Set Up Application as a Backend System

This example uses a Web application that responds to a request with a ping as a sample backend system. The Connectivity service supports HTTP and HTTPS as protocols and provides an easy way to consume REST-based Web services.

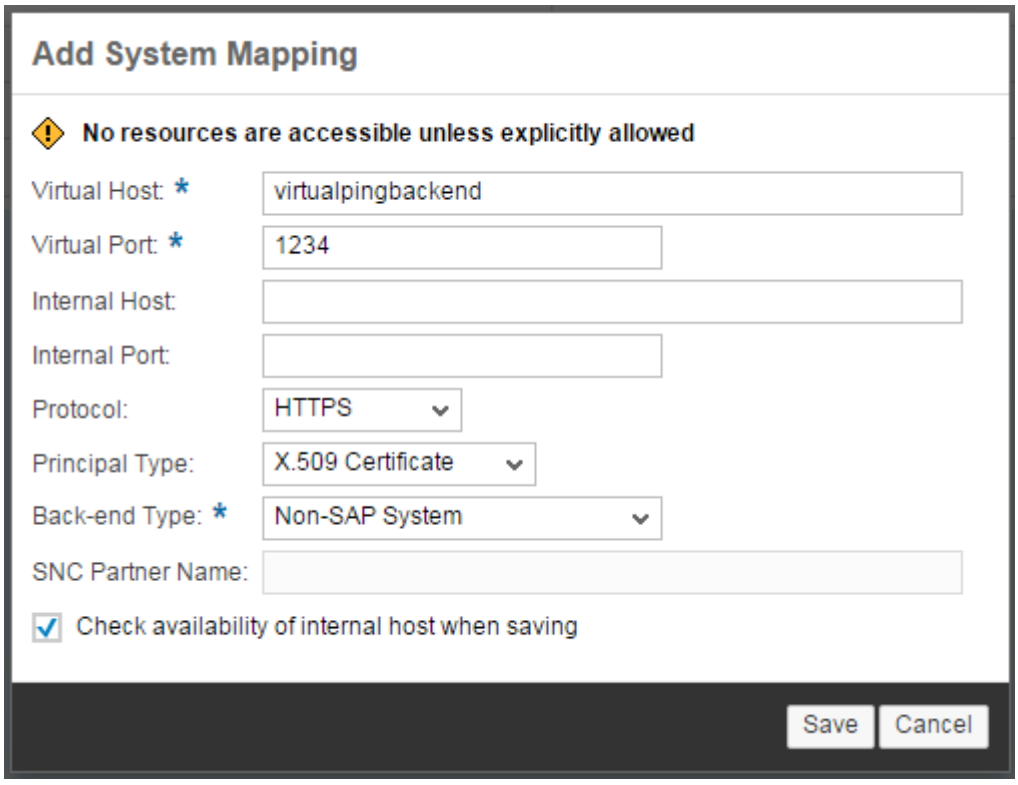
To set up the sample application as a backend system, see Set Up an Application as a Sample Backend System [page 190].

→ Tip

Instead of the sample backend system provided in this example, you can use other systems to be consumed through REST-based Web services.

Once the backend application is running on your local Tomcat, you need to configure the ping service, provided by the application, in your installed Cloud Connector. This is required since the Cloud Connector only allows access to trusted backend services. To do this, follow the steps below:

1. Open the Cloud Connector and from the *Content* navigation (in left), choose *Access Control*.
2. Under *Mapping Virtual To Internal System*, choose the *Add* button and define an entry as shown on the following screenshot. The *Internal Host* must be the physical host name of the machine on which the Tomcat of the backend application is running.



i Note

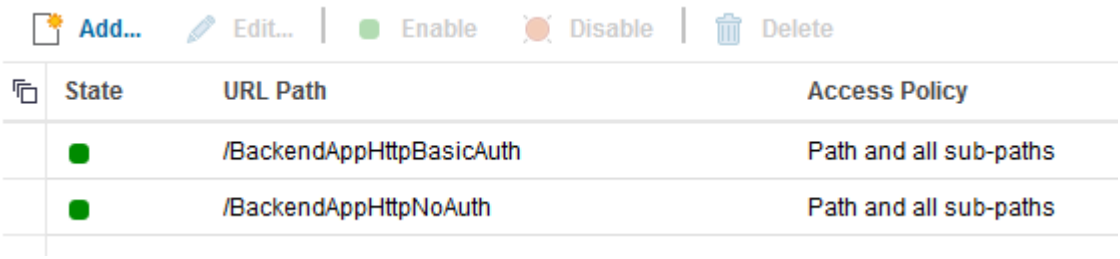
This step shows the procedure and screenshot for Cloud Connector versions prior to 2.9. For Cloud Connector versions as of 2.9.0, follow the steps in Configure Access Control (HTTP) [page 331] and enter the values shown in the screenshot above.

3. Choose *Save*. The newly mapped system appears in the table.
4. Click it. A new table, *Resources Accessible On <host>:<port>*, opens below.

i Note

For Cloud Connector versions as of 2.9.0, follow the steps in Configure Access Control (HTTP) [page 331], section *Limiting the Accessible Services for HTTP(S)*, and enter the values as shown in the next step.

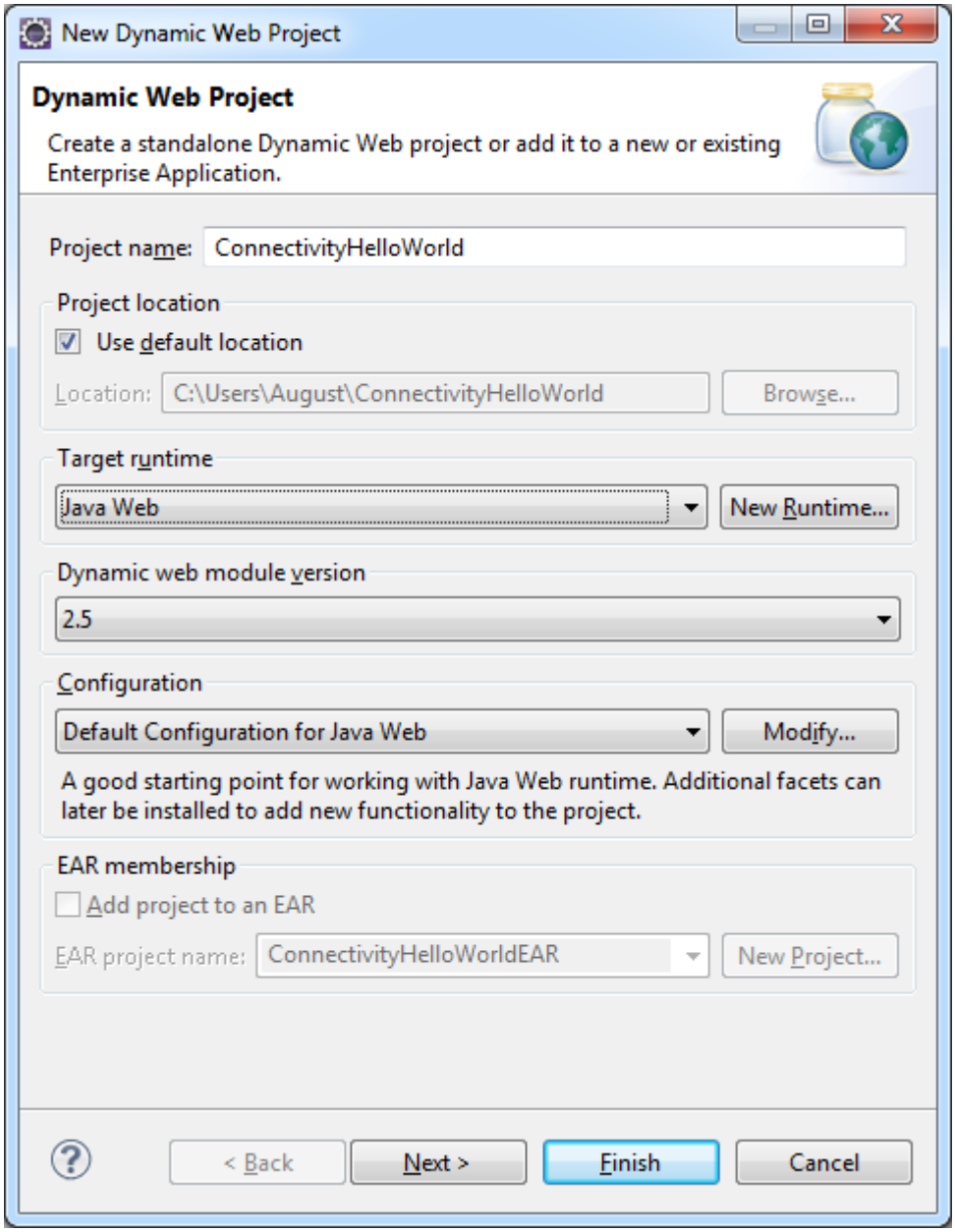
5. Specify the URL paths `/BackendAppHttpBasicAuth` and `/BackendAppHttpNoAuth` as accessible resources, as shown on the screenshot below. When defining the paths, make sure you have selected the *Path and all sub-paths* option.



> ### Note
>
> In case you use SDK with version equal to or lower than **`1.44.0.1`** (Java Web) and **`2.24.13`** (Java EE 6 Web Profile), you should find the WAR files in directory `<SDK_location>/tools/samples/connectivity/onpremise`, under the names **PingAppHttpNoAuth.war** and **PingAppHttpBasicAuth.war**. Also, the URL paths should be `/PingAppHttpBasicAuth` and `/PingAppHttpNoAuth`.

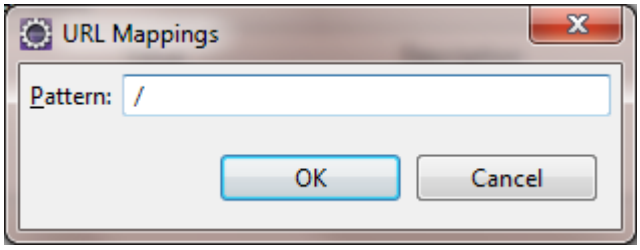
2. Create a Dynamic Web Project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. From the Eclipse main menu, choose *File* > *New* > *Dynamic Web Project* .
3. In the *Project name* field, enter **`ConnectivityHelloWorld`** .
4. In the *Target Runtime* pane, select the runtime you want to use to deploy the application. In this example, we choose **`Java Web`**.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to finalize the creation of your project.



3. Create a Sample Servlet

1. From the *ConnectivityHelloWorld* context menu, choose *New* > *Servlet*.
2. Enter `hello` as the *Java package* and `ConnectivityServlet` as the *Class name* and choose *Next*.
3. In the *URL mappings* field, select */ConnectivityServlet* and choose *Edit*.
4. In the *Pattern* field, replace the current value with just "`/`". In this way, the servlet will be mapped as a welcome page for the application.



5. Choose *Finish* so that the *ConnectivityServlet.java* servlet is created and opened in the Java editor.
6. Go to *ConnectivityHelloWorld* > *WebContent* > *WEB-INF* and open the `web.xml` file.
7. Add the following code block to the `<web-app>` element, respectively:

```
<resource-ref>
    <res-ref-name>outbound-internet-destination</res-ref-name>
    <res-type>com.sap.core.connectivity.api.http.HttpDestination</res-type>
</resource-ref>
```

```
<resource-ref>
    <res-ref-name>connectivity/DestinationFactory</res-ref-name>
    <res-type>com.sap.core.connectivity.api.DestinationFactory</res-type>
</resource-ref>
```

> **i Note**
>
> - Destinations **`backend-no-auth-destination`** and **`backend-basic-auth-destination`** will be looked-up via DestinationFactory JNDI lookup. For more information, see HttpDestination Library [page 147].
> - In case you use destinations as resource reference, the value of the `<res-ref-name>` element in the `web.xml` file should match the name of the destination that you want to be retrieved at runtime. In this case, the destination name is **`outbound-internet-destination`**.

8. Replace the entire servlet class to make use of the destination API. The destination API is visible by default for cloud applications and must not be added explicitly to the application class path.

```
package com.sap.cloud.sample.connectivity;

import java.io.IOException;
import java.io.InputStream;

import static java.net.HttpURLConnection.HTTP_OK;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.sap.core.connectivity.api.http.HttpDestination;
import com.sap.core.connectivity.api.DestinationFactory;
```

```
/**
 * Servlet class making HTTP calls to specified HTTP destinations.
 * Destinations are used in the following exemplary connectivity
scenarios:<br>
 * - Connecting to an outbound Internet resource using HTTP destinations<br>
 * - Connecting to an on-premise backend using on-premise HTTP
destinations,<br>
 *   where the destinations could have no authentication or basic
authentication.<br>
 *
 * * NOTE: The Connectivity
                                    service API is located under
 * <code>com.sap.core.connectivity.api</code>. The old API under
 * <code>com.sap.core.connectivity.httpdestination.api</code> has been
deprecated.
 */
public class ConnectivityServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final int COPY_CONTENT_BUFFER_SIZE = 1024;
    private static final Logger LOGGER =
LoggerFactory.getLogger(ConnectivityServlet.class);

    /** {@inheritDoc} */
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpClient httpClient = null;
        String destinationName = request.getParameter("destname");
        try {
            // Get HTTP destination
            Context ctx = new InitialContext();
            HttpDestination destination = null;
            if (destinationName != null) {
                DestinationFactory destinationFactory = (DestinationFactory)
ctx.lookup(DestinationFactory.JNDI_NAME);
                destination = (HttpDestination)
destinationFactory.getDestination(destinationName);
            } else {
                // The default request to the Servlet will use outbound-
internet-destination
                destinationName = "outbound-internet-destination";
                destination = (HttpDestination) ctx.lookup("java:comp/env/"
+  destinationName);
            }

            // Create HTTP client
            httpClient = destination.createHttpClient();

            // Execute HTTP request
            HttpGet httpGet = new HttpGet();
            HttpResponse httpResponse = httpClient.execute(httpGet);

            // Check response status code
            int statusCode = httpResponse.getStatusLine().getStatusCode();
            if (statusCode != HTTP_OK) {
                throw new ServletException("Expected response status code is
200 but it is " + statusCode + " .");
            }

            // Copy content from the incoming response to the outgoing
response
            HttpEntity entity = httpResponse.getEntity();
            if (entity != null) {
                InputStream instream = entity.getContent();
                try {
                    byte[] buffer = new byte[COPY_CONTENT_BUFFER_SIZE];
                    int len;
                    while ((len = instream.read(buffer)) != -1) {
```

```
                        response.getOutputStream().write(buffer, 0, len);
                    }
                } catch (IOException e) {
                    // In case of an IOException the connection will be
released
                    // back to the connection manager automatically
                    throw e;
                } catch (RuntimeException e) {
                    // In case of an unexpected exception you may want to
abort
                    // the HTTP request in order to shut down the underlying
                    // connection immediately.
                    httpGet.abort();
                    throw e;
                } finally {
                    // Closing the input stream will trigger connection
release
                    try {
                        instream.close();
                    } catch (Exception e) {
                    // Ignore
                    }
                }
            }
        } catch (NamingException e) {
            // Lookup of destination failed
            String errorMessage = "Lookup of destination failed with reason: "
                    + e.getMessage()
                    + ". See "
                    + "logs for details. Hint: Make sure to have the
destination "
                    + destinationName + " configured.";
            LOGGER.error("Lookup of destination failed", e);
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                    errorMessage);
        } catch (Exception e) {
            // Connectivity operation failed
            String errorMessage = "Connectivity operation failed with reason:
"
                    + e.getMessage()
                    + ". See "
                    + "logs for details. Hint: Make sure to have an HTTP
proxy configured in your "
                    + "local Eclipse environment in case your environment
uses "
                    + "an HTTP proxy for the outbound Internet "
                    + "communication.";
            LOGGER.error("Connectivity operation failed", e);
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                    errorMessage);
        } finally {
            // When HttpClient instance is no longer needed, shut down the
connection manager to ensure immediate
            // deallocation of all system resources
            if (httpClient != null) {
                httpClient.getConnectionManager().shutdown();
            }
        }
    }
}
```

> ### i Note
>
> The given servlet can be run with different destination scenarios, for which user should specify the destination name as a requested parameter in the calling URL. In the case of on-premise connection to a backend system, the destination name should be either **`backend-basic-auth-destination`** or

backend-no-auth-destination, depending on the chosen authentication type scenario. For example: `<application_URL>/?destname=backend-no-auth-destination`

9. Save the Java editor and make sure the project compiles without errors.

4. Deploy the Application

> ⚠ Caution
>
> - If you use SDK for Java Web, we just recommend that you create a destination before starting the application.
> - If you use SDK for Java EE 6 Web Profile, you must create a destination **before** starting the application.

1. To deploy your Web application locally or on the cloud, follow the steps described in the respective pages:

2. Once the application is deployed successfully on a local server and on the cloud, the application issues an exception. This exception says that destination **backend-basic-auth-destination** or **backend-no-auth-destination** has not been specified yet:

```
HTTP Status 500 - Connectivity operation failed with reason: Destination with
name backend-no-auth-destination cannot be found. Make sure it is created and
configured.. See logs for details.
2014 01 10 08:11:01#
+00#ERROR#com.sap.cloud.sample.connectivity.ConnectivityServlet##anonymous#htt
p-bio-8041-exec-1##conngold#testsample#web#null#null#Connectivity operation
failed
com.sap.core.connectivity.api.DestinationNotFoundException: Destination with
name backend-no-auth-destination cannot be found. Make sure it is created and
configured.
    at
com.sap.core.connectivity.destinations.DestinationFactory.getDestination(Desti
nationFactory.java:20)
    at
com.sap.core.connectivity.cloud.destinations.CloudDestinationFactory.getDestin
ation(CloudDestinationFactory.java:28)
    at
com.sap.cloud.sample.connectivity.ConnectivityServlet.doGet(ConnectivityServle
t.java:50)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:735)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:848)
    at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFi
lterChain.java:305)
    at
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChai
n.java:210)
    at
com.sap.core.communication.server.CertValidatorFilter.doFilter(CertValidatorFi
lter.java:321)
    at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFi
lterChain.java:243)
    ...
```

3. As a next step, you need to configure **backend-no-auth-destination** or **backend-basic-auth-destination**.

For more information, see HttpDestination Library [page 147].

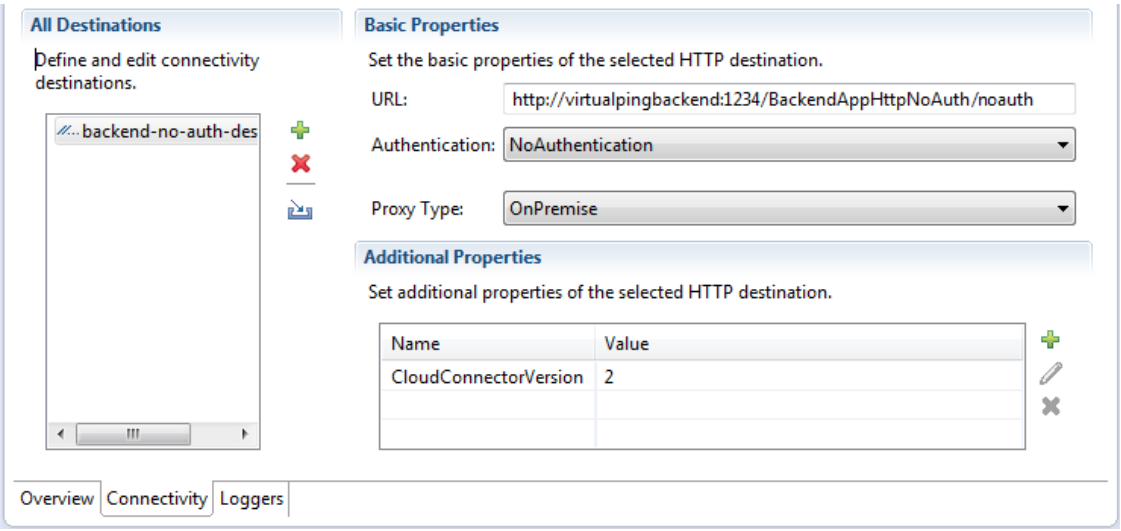
5. Configure the Destination in the Cloud

To configure the destination in SAP BTP, you need to use the virtual host name (**`virtualpingbackend`**) and port (**`1234`**) specified in one of the previous steps on the Cloud Connector's *Access Control* tab page.

> **i Note**
>
> On-premise destinations support HTTP connections only. Thus, when defining a destination in the SAP BTP cockpit, always enter the URL as **http***://virtual.host:virtual.port*, even if the backend requires an HTTPS connection.
>
> The connection from an SAP BTP application to the Cloud Connector (through the tunnel) is encrypted with TLS anyway. There is no need to "double-encrypt" the data. Then, for the leg from the Cloud Connector to the backend, you can choose between using HTTP or HTTPS. The Cloud Connector will establish an SSL/TLS connection to the backend, if you choose HTTPS.

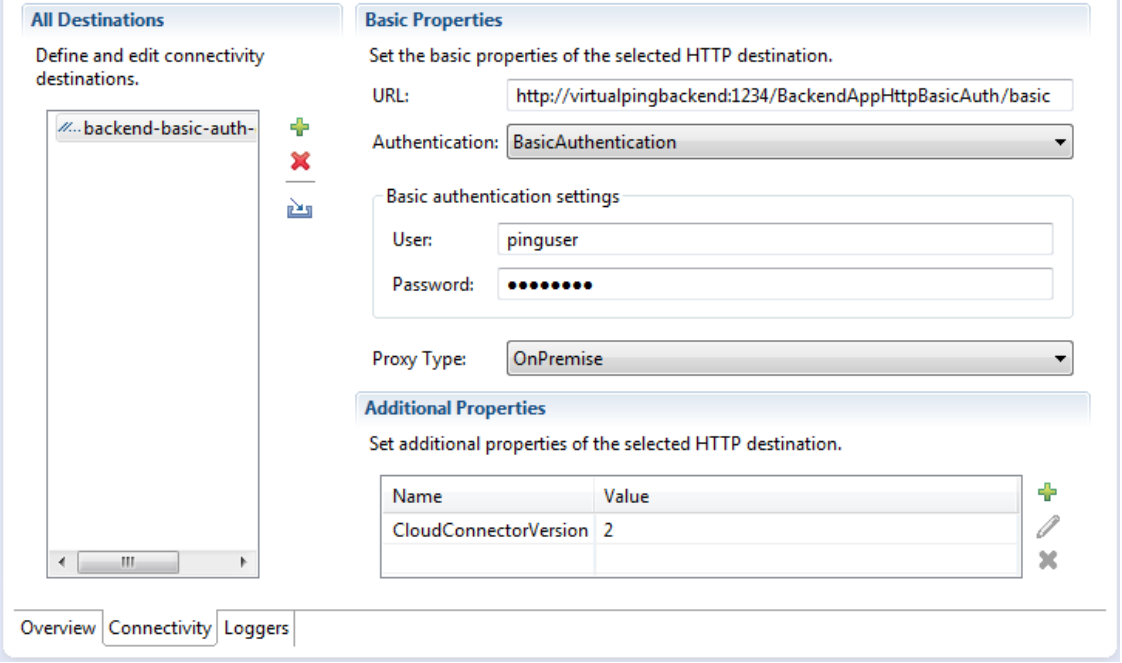
1. In the Eclipse IDE, open the *Servers* view and double-click on `<application>.<subaccount>` to open the SAP BTP editor.
2. Open the *Connectivity* tab page.
3. In the *All Destinations* section, choose ✚ to create a new destination with the name **`backend-no-auth-destination`** or **`backend-basic-auth-destination`**.
   - To connect with no authentication, use the following configuration:

```
Name=backend-no-auth-destination
Type=HTTP
URL=http://virtualpingbackend:1234/BackendAppHttpNoAuth/noauth
Authentication=NoAuthentication
ProxyType=OnPremise
CloudConnectorVersion=2
```



   - To connect with basic authentication, use the following configuration:

```
Name=backend-basic-auth-destination
Type=HTTP
URL=http://virtualpingbackend:1234/BackendAppHttpBasicAuth/basic
Authentication=BasicAuthentication
User=pinguser
Password=pingpassword
ProxyType=OnPremise
CloudConnectorVersion=2
```



4. Save the destination.
5. The *Connectivity* editor automatically saves the configuration in SAP BTP.
6. Call the URL that references the cloud application again in the Web browser. The application should now return the ping response.

Next Step

You can monitor the state and logs of your Web application deployed on SAP BTP.

For more information, see .

1.4.4.1.2.3.4 Consume Backend Systems (Java Web Tomcat 7)

Context

This example demonstrates how a sample Web application consumes a backend system via HTTP(S) by using the Connectivity service. For simplicity, instead of using a real backend system, we use a second sample Web application containing `BackendServlet`. It mimics the backend system and can be called via HTTP(S).

The servlet code, the `web.xml` content, and the destination file (**`backend-no-auth-destination`**) used in this example are mapped to the connectivity sample project located in `<SDK_location>/samples/connectivity`. You can directly import this sample in your Eclipse IDE. For more information, see .

The example includes the following sections:

1. Set Up Application as a Backend System [page 182]
2. Create a Dynamic Web Project [page 196]
3. Create a Sample Servlet [page 185]
4. Deploy the Application [page 199]
5. Configure the Destination in the Cloud [page 188]

Connectivity User Roles

In the on-demand to on-premise connectivity end-to-end scenario, different user roles are involved. The particular steps for the relevant roles are described below:

- *IT Administrator* - Sets up and configures the Cloud Connector. Scenario steps:
  1. Downloads the Cloud Connector from https://tools.hana.ondemand.com/#cloud
  2. Installs the connector.
  3. Establishes an SSL tunnel from the connector to an SAP BTP.
  4. Configures the exposed backend systems and resources.
- *Application Developer* - Develops Web applications using destinations. Scenario steps:
  1. Installs the Eclipse IDE, SAP BTP Tools for Java, and SDK.
  2. Develops a Java EE application using the destination API.
  3. Configures connectivity destinations as resources in the `web.xml` file.
  4. Configures connectivity destinations via the SAP BTP server adapter in Eclipse IDE.
  5. Deploys the Java EE application locally and on the cloud.
- *Subaccount Operator* - Deploys Web applications, configures their destinations, and conducts tests. Scenario steps:
  1. Obtains a ready Java EE application WAR file.
  2. Deploys the Java EE application to an SAP BTP subaccount.
  3. Uploads the connectivity destination configuration via the console client.
  4. Tests the Java EE application on a local server and deploys it again to a SAP BTP subaccount.

For more information, see Cloud Connector [page 232].

Prerequisites

- You have downloaded and configured the Cloud Connector. For more information, see Cloud Connector [page 232].
- You have downloaded and set up your Eclipse IDE, SAP BTP Tools for Java, and SDK.
  For more information, see Setting Up the Development Environment [page 845].

> **i Note**
>
> You need to install **`SDK for Java Web Tomcat 7`**.

1. Set Up Application as a Backend System

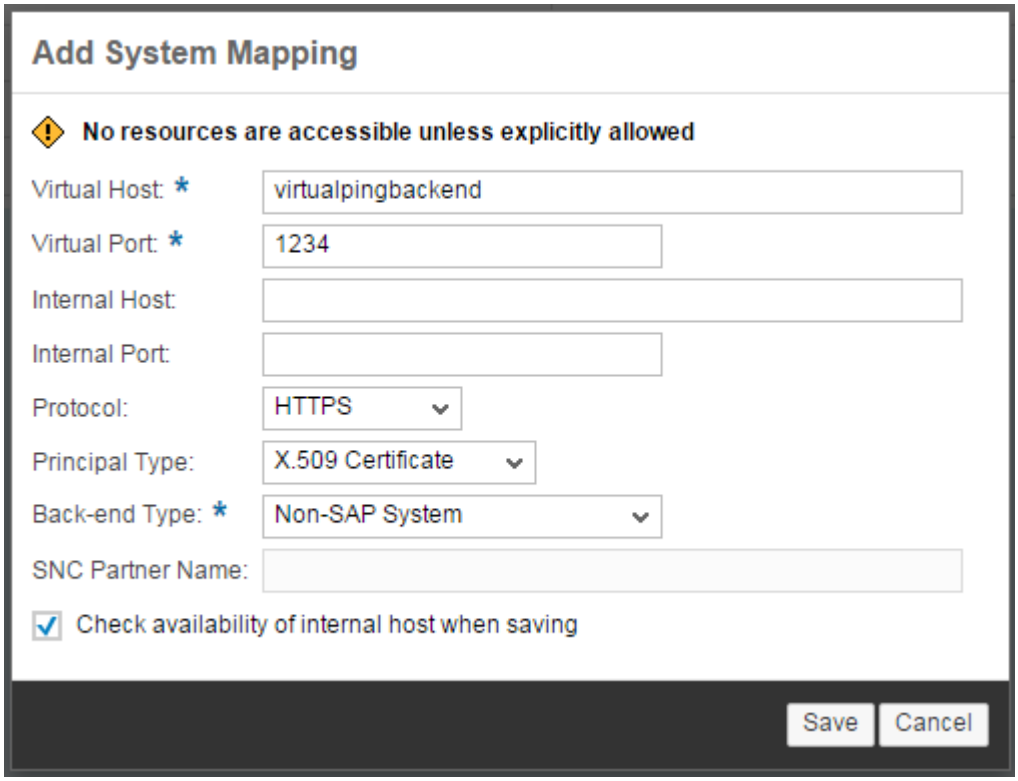
This example uses a Web application that responds to a request with a ping as a sample backend system. The Connectivity service supports HTTP and HTTPS as protocols and provides an easy way to consume REST-based Web services.

To set up the sample application as a backend system, see Set Up an Application as a Sample Backend System [page 190].

> **→ Tip**
>
> Instead of the sample backend system provided in this example, you can use other systems to be consumed through REST-based Web services.

Once the backend application is running on your local Tomcat, you need to configure the ping service, provided by the application, in your installed Cloud Connector. This is required since the Cloud Connector only allows access to white-listed backend services. To do this, follow the steps below:

1. Open the Cloud Connector and from the *Content* navigation (in left), choose *Access Control*.
2. Under *Mapping Virtual To Internal System*, choose the *Add* button and define an entry as shown on the following screenshot. The *Internal Host* must be the physical host name of the machine on which the Tomcat of the backend application is running.

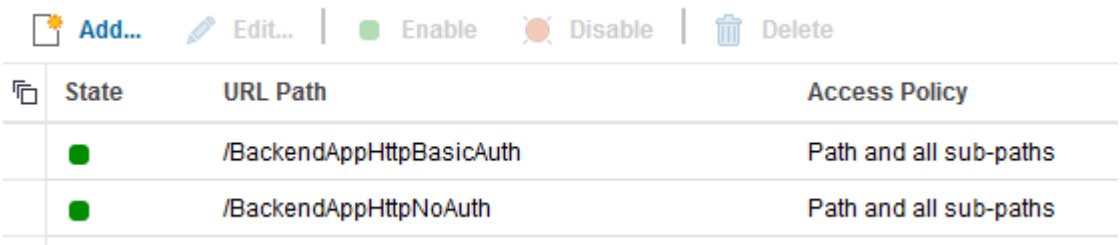


> **i Note**
>
> This step shows the procedure and screenshot for Cloud Connector versions prior to 2.9. For Cloud Connector versions as of 2.9.0, follow the steps in Configure Access Control (HTTP) [page 331] and enter the values shown in the screenshot above.

3. Choose *Save*. The newly mapped system appears in the table.
4. Click it. A new table, *Resources Accessible On <host>:<port>*, opens below.

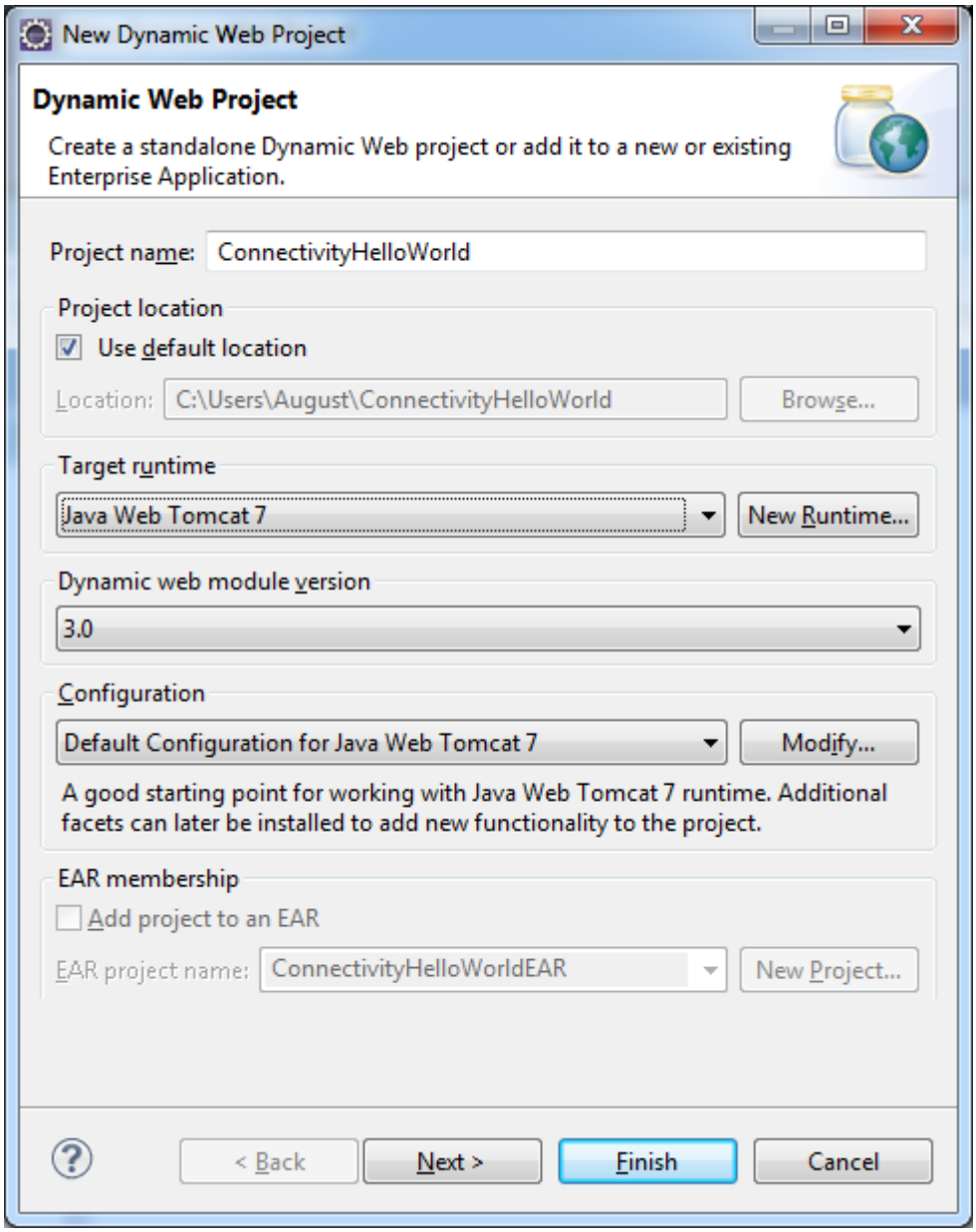
> **i Note**
>
> For Cloud Connector versions as of 2.9.0, follow the steps in Configure Access Control (HTTP) [page 331], section *Limiting the Accessible Services for HTTP(S)*, and enter the values as shown in the next step.

5. Specify the URL paths `/BackendAppHttpBasicAuth` and `/BackendAppHttpNoAuth` as accessible resources, as shown on the screenshot below. When defining the paths, make sure you have selected the *Path and all sub-paths* option.



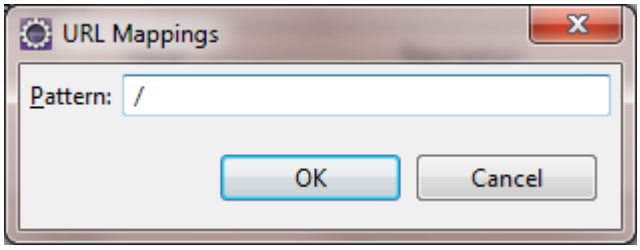
2. Create a Dynamic Web Project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. From the Eclipse main menu, choose *File* > *New* > *Dynamic Web Project*.
3. In the *Project name* field, enter `ConnectivityHelloWorld` .
4. In the *Target Runtime* pane, select `Java Web Tomcat 7` as the runtime you want to use to deploy the application.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to finalize the creation of your project.



3. Create a Sample Servlet

1. From the *ConnectivityHelloWorld* context menu, choose *New* > *Servlet*.
2. Enter **`hello`** as the *Java package* and **`ConnectivityServlet`** as the *Class name* and choose *Next*.
3. In the *URL mappings* field, select */ConnectivityServlet* and choose *Edit*.
4. In the *Pattern* field, replace the current value with just "**`/`**". In this way, the servlet will be mapped as a welcome page for the application.



5. Choose *Finish* so that the *ConnectivityServlet.java* servlet is created and opened in the Java editor.
6. Go to *ConnectivityHelloWorld* > *WebContent* > *WEB-INF* and open the `web.xml` file.
7. To consume connectivity configuration using JNDI, you need to define the `ConnectivityConfiguration` API as a resource in the `web.xml` file. Below is an example of a `ConnectivityConfiguration` resource, named **`connectivityConfiguration`**.

```
 <resource-ref>
    <res-ref-name>connectivityConfiguration</res-ref-name>
    <res-
type>com.sap.core.connectivity.api.configuration.ConnectivityConfiguration</
res-type>
 </resource-ref>
```

> ### Note
>
> Destination **`backend-no-auth-destination`** will be looked-up via ConnectivityConfiguration JNDI lookup. For more information, see ConnectivityConfiguration API [page 139].

8. Replace the entire servlet class to make use of the configuration API. The configuration API is visible by default for cloud applications and must not be added explicitly to the application class path.

```
package com.sap.cloud.sample.connectivity;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.InetSocketAddress;
import java.net.Proxy;
import java.net.URL;

import javax.annotation.Resource;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import com.sap.cloud.account.TenantContext;
import com.sap.core.connectivity.api.configuration.ConnectivityConfiguration;
import com.sap.core.connectivity.api.configuration.DestinationConfiguration;
/**
 * Servlet class making http calls to specified http destinations.
 * Destinations are used in the following example connectivity scenarios:<br>
 * - Connecting to an outbound Internet resource using HTTP destinations<br>
 * - Connecting to an on-premise backend using on premise HTTP
destinations,<br>
 *   where the destinations have no authentication.<br>
 */
public class ConnectivityServlet extends HttpServlet {
    @Resource
    private TenantContext  tenantContext;

    private static final long serialVersionUID = 1L;
    private static final int COPY_CONTENT_BUFFER_SIZE = 1024;
    private static final Logger LOGGER =
LoggerFactory.getLogger(ConnectivityServlet.class);

    private static final String ON_PREMISE_PROXY = "OnPremise";

    /** {@inheritDoc} */
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpURLConnection urlConnection = null;
        String destinationName = request.getParameter("destname");

        // The default request to the Servlet will use outbound-internet-
destination
        if (destinationName == null) {
            destinationName = "outbound-internet-destination";
        }

        try {
            // Look up the connectivity configuration API
            Context ctx = new InitialContext();
            ConnectivityConfiguration configuration =
(ConnectivityConfiguration) ctx.lookup("java:comp/env/
connectivityConfiguration");

            // Get destination configuration for "destinationName"
            DestinationConfiguration destConfiguration =
configuration.getConfiguration(destinationName);
            if (destConfiguration == null) {

response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                        String.format("Destination %s is not found. Hint:
Make sure to have the destination configured.", destinationName));
                return;
            }

            // Get the destination URL
            String value = destConfiguration.getProperty("URL");
            URL url = new URL(value);

            String proxyType = destConfiguration.getProperty("ProxyType");
            Proxy proxy = getProxy(proxyType);

            urlConnection = (HttpURLConnection) url.openConnection(proxy);

            // Insert the required header in the request for on-premise
destinations
            injectHeader(urlConnection, proxyType);
```

```
            // Copy content from the incoming response to the outgoing
response
            InputStream instream = urlConnection.getInputStream();
            OutputStream outstream = response.getOutputStream();
            copyStream(instream, outstream);
        } catch (Exception e) {
            // Connectivity operation failed
            String errorMessage = "Connectivity operation failed with reason:
"
                    + e.getMessage()
                    + ". See "
                    + "logs for details. Hint: Make sure to have an HTTP
proxy configured in your "
                    + "local environment in case your environment uses "
                    + "an HTTP proxy for the outbound Internet "
                    + "communication.";
            LOGGER.error("Connectivity operation failed", e);
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                    errorMessage);
        }
    }

    private Proxy getProxy(String proxyType) {
        String proxyHost = null;
        int proxyPort;
        if (ON_PREMISE_PROXY.equals(proxyType)) {
            // Get proxy for on-premise destinations
            proxyHost = System.getenv("HC_OP_HTTP_PROXY_HOST");
            proxyPort =
Integer.parseInt(System.getenv("HC_OP_HTTP_PROXY_PORT"));
        } else {
            // Get proxy for internet destinations
            proxyHost = System.getProperty("http.proxyHost");
            proxyPort =
Integer.parseInt(System.getProperty("http.proxyPort"));
        }
        return new Proxy(Proxy.Type.HTTP, new InetSocketAddress(proxyHost,
proxyPort));
    }

    private void injectHeader(HttpURLConnection urlConnection, String
proxyType) {
        if (ON_PREMISE_PROXY.equals(proxyType)) {
            // Insert header for on-premise connectivity with the consumer
subaccount name
            urlConnection.setRequestProperty("SAP-Connectivity-
ConsumerAccount", tenantContext.getAccountName());
        }
    }

    private void copyStream(InputStream inStream, OutputStream outStream)
throws IOException {
        byte[] buffer = new byte[COPY_CONTENT_BUFFER_SIZE];
        int len;
        while ((len = inStream.read(buffer)) != -1) {
            outStream.write(buffer, 0, len);
        }
    }
}
```

> ### i Note
>
> The given servlet can be run with different destination scenarios, for which user should specify the destination name as a requested parameter in the calling URL. In the case of on-premise connection to

a backend system, the destination names should be **backend-no-auth-destination**. That is, it will be accessed at: `<application_URL>/?destname=backend-no-auth-destination`

> **i Note**
>
> When accessing a destination with a specific authentication type, use AuthenticationHeaderProvider API [page 141] to get authentication headers and then inject them in all requests to this destination.

9. Save the Java editor and make sure the project compiles without errors.

4. Deploy the Application

> **i Note**
>
> We only recommend but not obligate that you create the destination before starting the application.

1. To deploy your Web application locally or on the cloud, follow the steps described in the respective pages:

2. Once the application is successfully deployed locally or on the cloud, the application issues an exception saying that the **backend-no-auth-destination** destination has not been specified yet:

```
HTTP Status 500 - Destination backend-no-auth-destination is not found. Hint:
Make sure to have the destination configured.
```

3. As a next step, you need to configure **backend-no-auth-destination**.

For more information, see ConnectivityConfiguration API [page 139].

5. Configure the Destination in the Cloud

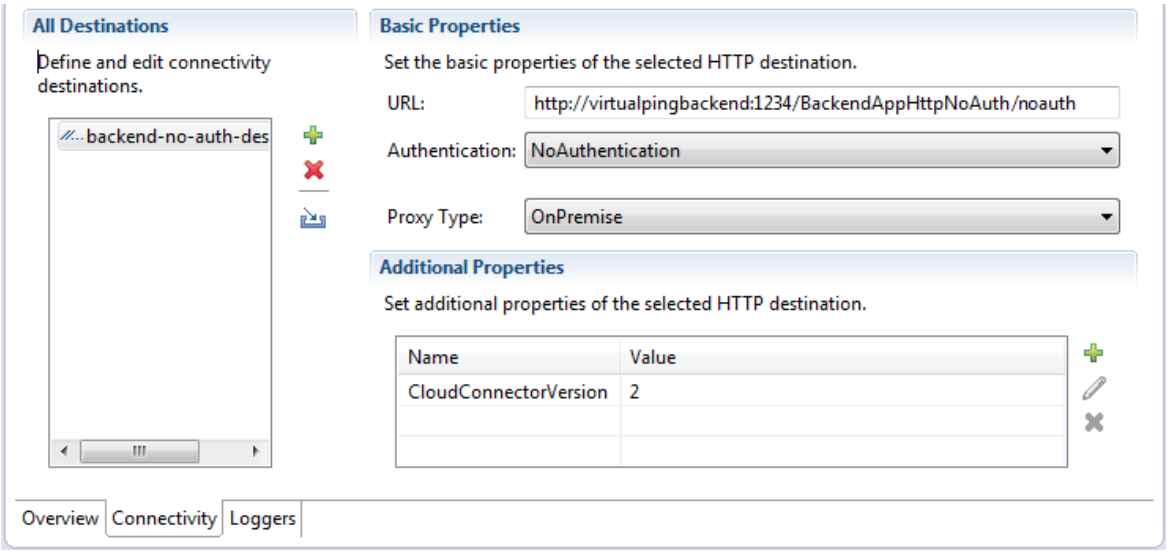
To configure the destination in SAP BTP, you need to use the virtual host name (**virtualpingbackend**) and port (**1234**) specified in one of the previous steps on the Cloud Connector's *Access Control* tab page.

> **i Note**
>
> - On-premise destinations support HTTP connections only.
> - The connection from an application to the Cloud Connector (through the tunnel) is encrypted on TLS level. Also, you can choose between using HTTP or HTTPS to hop from the Cloud Connector to the back end.

1. In the Eclipse IDE, open the *Servers* view and double-click `<application>.<subaccount>` to open the cloud server editor.
2. Open the *Connectivity* tab page.
3. In the *All Destinations* section, choose  to create a new destination with the name **backend-no-auth-destination**.

4. Use the following configuration:

```
Name=backend-no-auth-destination
Type=HTTP
URL=http://virtualpingbackend:1234/BackendAppHttpNoAuth/noauth
Authentication=NoAuthentication
ProxyType=OnPremise
CloudConnectorVersion=2
```



5. Save the destination.
6. The *Connectivity* editor automatically saves the configuration in the cloud.
7. Call the URL that references the cloud application again in the internal Web browser. The application should now return the ping response.

Next Step

You can monitor the state and logs of your Web application deployed on SAP BTP.

For more information, see .

Related Information

JavaDoc ConnectivityConfiguration

JavaDoc DestinationConfiguration

JavaDoc AuthenticationHeaderProvider

AuthenticationHeaderProvider API [page 141]

1.4.4.1.2.3.5 Set Up an Application as a Sample Backend System

Overview

This section describes how you set up a simple ping Web application that is used as a backend system.

Prerequisites

You have downloaded SAP BTP SDK on your local file system.

Procedure

1. Set up a servlet container such as Tomcat .
2. Add a user and role for basic authentication by adding the following lines to the*tomcat-users.xml* file in directory `<TOMCAT_HOME>/conf` file:

```
<role rolename="pingrole"/>
<user name="pinguser" password="pingpassword" roles="pingrole" />
```

3. From the SDK location, go to `/samples/connectivity/onpremise`, copy files *BackendAppHttpNoAuth.war* and *BackendAppHttpBasicAuth.war* and paste them into the `<TOMCAT_HOME>/webapps` directory.
4. Start Tomcat and access the on-premise applications at the URLs below. Use **`pinguser`** / **`pingpassword`** as the credentials.
   - `http://localhost:8080/BackendAppHttpNoAuth/noauth`
   - `http://localhost:8080/BackendAppHttpBasicAuth/basic`

> **i Note**
>
> In case you use SDK with version equal to or lower than, respectively, **`1.44.0.1`** (Java Web) and **`2.24.13`** (Java EE 6 Web Profile), you should find the WAR files in directory `<SDK_location>/tools/samples/connectivity/onpremise`, under the names **PingAppHttpNoAuth.war** and **PingAppHttpBasicAuth.war**. Also, you should access the applications at the relevant URLs:
>
> - `http://localhost:8080/PingAppHttpNoAuth/pingnoauth`
> - `http://localhost:8080/PingAppHttpBasicAuth/pingbasic`

Related Information

Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170]

1.4.4.1.3 Invoking ABAP Function Modules via RFC

Call a remote-enabled function module in an on-premise ABAP server from your Neo application, using the RFC protocol.

Find the tasks and prerequisites that are required to consume an on-premise ABAP function module via RFC, using the Java Connector (JCo) API as a built-in feature of SAP BTP.

Tasks

Task Type	Task
 Operator	Prerequisites [page 191]
 Operator and/or Developer	About JCo [page 192]
	Installation Prerequisites for JCo Applications [page 192]
	Consume Connectivity via RFC [page 193]
	Restrictions [page 193]

Prerequisites



Before you can use RFC communication for an SAP BTP application, you must configure:

- A destination on SAP BTP to use RFC.
  For more information, see Configure Destinations from the Console Client [page 56] and RFC Destinations [page 113].
- RFC connectivity between a backend system and the application. To do this, you must install the Cloud Connector [page 232] in your internal network and configure it to expose a remote-enabled function module in an on-premise ABAP system.

For more information, see Initial Configuration (RFC) [page 287] and Configure Access Control (RFC) [page 338].

Back to Tasks [page 191]

About JCo



To learn in detail about the SAP JCo API, see the JCo 3.0 documentation on SAP Support Portal.

> **i** Note
>
> Some sections of this documentation are not applicable to SAP BTP:
>
> - **Architecture**: CPIC is only used in the last mile from your Cloud Connector to the back end. From SAP BTP to the Cloud Connector, TLS-protected communication is used.
> - **Installation**: SAP BTP runtimes already include all required artifacts.
> - **Customizing and Integration**: On SAP BTP, the integration is already done by the runtime. You can concentrate on your business application logic.
> - **Server Programming**: The programming model of JCo on SAP BTP does not include server-side RFC communication.
> - **IDoc Support for External Java Applications**: Currently, there is no IDocLibrary for JCo available on SAP BTP

Back to Tasks [page 191]

Installation Prerequisites for JCo Applications



To develop a JCo application:

- your SDK version must be `1.29.18` (*SDK Java Web*), or `2.11.6` (*SDK for Java EE 6 Web Profile*).
- your SDK local runtime must be hosted by a 64-bit JVM. SDKs of Tomcat 7, Tomcat 8, and TomEE 7 runtime support JCo from the very beginning.
- on Windows platforms, you must install **Microsoft Visual Studio C++ 2013 runtime libraries (vcredist_x64.exe)**. To download this package, go to https://www.microsoft.com/en-us/download/details.aspx?id=40784 .

Back to Tasks [page 191]

Consume Connectivity via RFC



You can call a service from a fenced customer network using a simple application which consumes a simple on-premise, remote-enabled function module.

Invoking function modules via RFC is enabled by a JCo API that is comparable to the one available in SAP NetWeaver Application Server Java (version 7.10+), and in JCo standalone 3.0. If you are an experienced JCo developer, you can easily develop a Web application using JCo: you simply consume the APIs like you do in other Java environments. Restrictions that apply in the cloud environment are mentioned in the **Restrictions** section below.

Find a sample Web application in Invoke ABAP Function Modules in On-Premise ABAP Systems [page 194].

Back to Tasks [page 191]

Restrictions



- **JCoServer** functionality cannot be used within SAP BTP.
- **Environment embedding**, such as offered by JCo standalone 3.0, is not possible. This is, however, similar to SAP NetWeaver AS Java.
- By default, a **stateful sequence of function module invocations** must happen in a single HTTP request/response cycle.
  To make a stateful sequence work across several request/response cycles, you must set the parameter `advancedJCoUsage`. In the cloud cockpit, navigate to your subaccount, and from the left-side subaccount menu choose *Applications* > *Java Applications* > *Deploy Application*. In the pop-up window, enter **`-DadvancedJCoUsage=true`** in the field `<JVM Arguments>`. Alternatively, you can specify the argument when deploying with the Neo SDK tools.

  > **i** Note
  >
  > To add the parameter to an existing application, select the application and choose *Update*. When you are done, you must restart the application.

  The minimal runtime versions for supporting this capability are listed below:

Runtime/SDK	Minimal Version
Java Web	1.171.11.1
Java Web Tomcat 7	2.118.20.1
Java Web Tomcat 8	3.73.18.1
Java EE 6 Web Profile	2.153.9.1
Java EE 7 Web Profile TomEE 7	1.42.19.1

- **Logon authentication** only supports user/password credentials (basic authentication) and principal propagation. See Create RFC Destinations [page 81] and User Logon Properties [page 113].
- **Provider/subscription model** for applications is only fully supported in newer runtime versions. If you still want to use it in older ones, you need to make sure that destinations are named differently in all accounts. Minimal runtime versions for full support are listed below:

Runtime/SDK	Minimal Version
Java Web	1.142.0
Java Web Tomcat 7	2.89.13
Java Web Tomcat 8	3.43.12
Java EE 6 Web Profile	2.124.0
Java EE 7 Web Profile TomEE 7	1.13.12

- The supported set of **configuration properties** is restricted. For details, see RFC Destinations [page 113].

Back to Tasks [page 191]

Related Information

Java Connector API [page 141]

1.4.4.1.3.1 Invoke ABAP Function Modules in On-Premise ABAP Systems

Context

This example shows how a sample Web application invokes a function module in an on-premise ABAP system via RFC by using theConnectivity service.

The procedure contains the following sections:

- Presenting the user roles
- Defining the installation prerequisites
- Developing a sample Web application that uses theConnectivity service to consume the simple function module `STFC_CONNECTION`.

Connectivity User Roles

Different user roles are involved in the on-demand to on-premise connectivity end-to-end scenario. The particular steps for the relevant roles are described below:

IT Administrator

This role sets up and configures the Cloud Connector. Scenario steps:

1. Downloads the Cloud Connector from https://tools.hana.ondemand.com/#cloud
2. Installs the Cloud Connector.
3. Establishes an SSL tunnel from the connector to an SAP BTP subaccount.
4. Configures the exposed back-end systems and resources.

Application Developer

This role develops Web applications using destinations. Scenario steps:

1. Installs the Eclipse IDE, SAP BTP Tools for Java, and SDK.
2. Develops a Java EE application using the destination API.
3. Configures connectivity destinations as resources in the *web.xml* file.
4. Configures connectivity destinations via the SAP BTP server adapter in Eclipse IDE.
5. Deploys the Java EE application locally and on the cloud.

Subaccount Operator

This role deploys Web applications, configures their destinations, and conducts tests. Scenario steps:

1. Obtains a ready Java EE application WAR file.
2. Deploys the Java EE application in an SAP BTP subaccount.
3. Uploads the connectivity destination configuration via the console client.
4. Tests the Java EE application on a local server and deploys it again to a SAP BTP subaccount.

Installation Prerequisites

- You have downloaded and set up your Eclipse IDE and SAP BTP Tools for Java.
- You have downloaded the SDK. Its version needs to be at least `1.29.18` (*SDK for Java Web*), `2.11.6` (*SDK for Java EE 6 Web Profile*), or `2.9.1` (*SDK for Java Web Tomcat 7*), respectively.
- Your local runtime needs to be hosted by a 64-bit JVM. On Windows platforms, you need to install *Microsoft Visual C++ 2010 Redistributable Package (x64)*.
- You have downloaded and configured your Cloud Connector. Its version needs to be at least `1.3.0`.

To download the SAP tools, go to https://tools.hana.ondemand.com/#cloud.

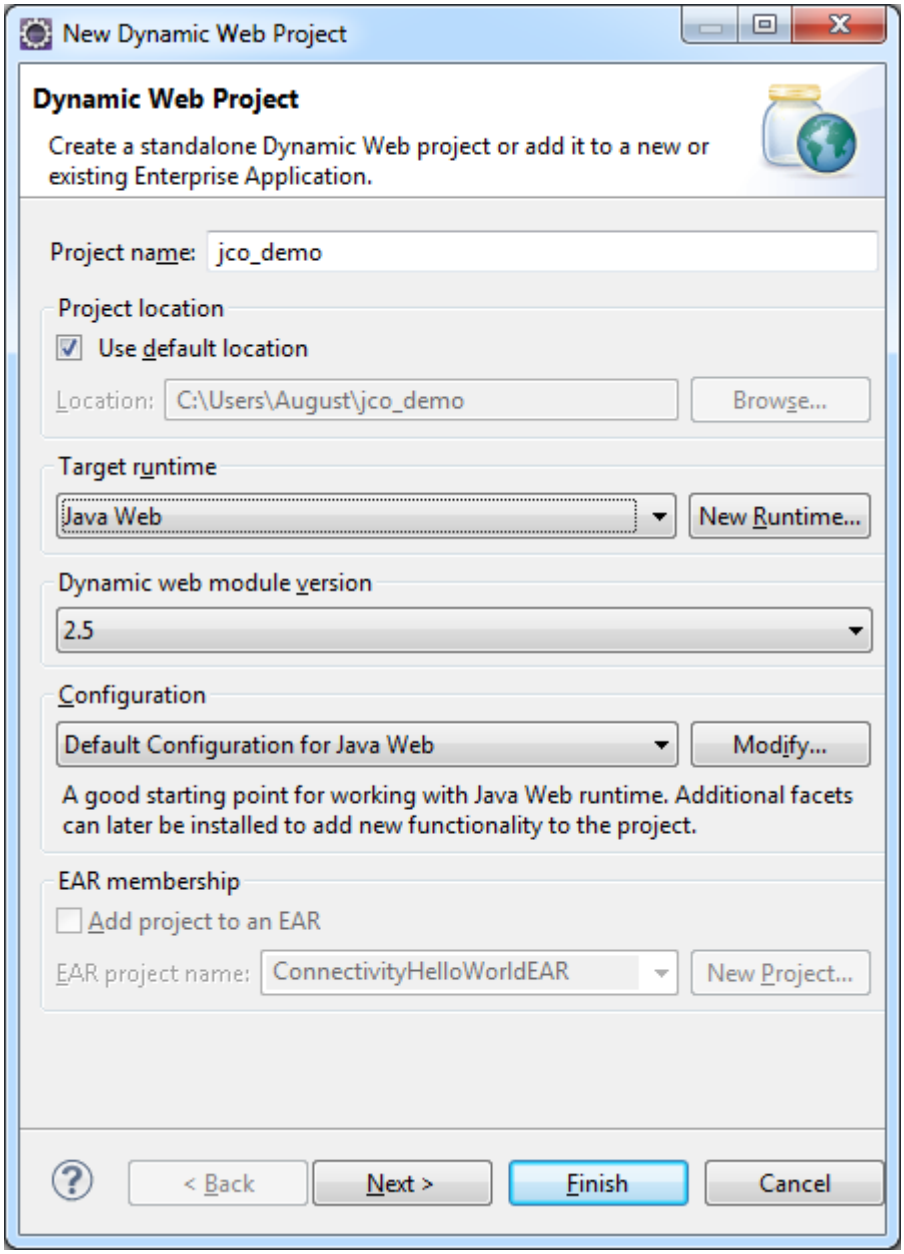
To download the Microsoft Visual C++ package, go to http://www.microsoft.com/en-us/download/details.aspx?id=14632 .

To read the installation documentation, go to Setting Up the Development Environment [page 845] and Installation [page 238].

Creating a Dynamic Web Project

Procedure

1. In the Eclipse IDE, open the *Java EE* perspective.
2. From the Eclipse main menu, choose *New* > *Dynamic Web Project*.
3. In the *Project name* field, enter `jco_demo` .
4. In the *Target Runtime* pane, select the runtime you want to use to deploy the HelloWorld application. In this example, we choose `Java Web`.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to complete the creation of your project.



Creating a Sample Servlet

Procedure

1. From the *jco_demo* context menu, choose *New* > *Servlet* .
2. Enter `com.sap.demo.jco` as the *Java package* and `ConnectivityRFCExample` as the *Class name* and choose *Next*.

3. Choose *Finish* so that the *ConnectivityRFCExample.java* servlet is created and opened in the Java editor.
4. Replace the entire servlet class to make use of the JCo API. The JCo API is visible by default for cloud applications and must not be added explicitly to the application class path.

```
package com.sap.demo.jco;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.sap.conn.jco.AbapException;
import com.sap.conn.jco.JCoDestination;
import com.sap.conn.jco.JCoDestinationManager;
import com.sap.conn.jco.JCoException;
import com.sap.conn.jco.JCoFunction;
import com.sap.conn.jco.JCoParameterList;
import com.sap.conn.jco.JCoRepository;
/**
 * Sample application that uses the Connectivity
                                    service. In particular,
 * it makes use of the capability to invoke a function module in an ABAP
system
 * via RFC
 *
 * Note: The JCo APIs are available under <code>com.sap.conn.jco</code>.
 */
public class ConnectivityRFCExample extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    public ConnectivityRFCExample()
    {
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException
    {
        PrintWriter responseWriter = response.getWriter();
        try
        {
            // access the RFC Destination "JCoDemoSystem"
            JCoDestination
destination=JCoDestinationManager.getDestination("JCoDemoSystem");
            // make an invocation of STFC_CONNECTION in the backend;
            JCoRepository repo=destination.getRepository();
            JCoFunction stfcConnection=repo.getFunction("STFC_CONNECTION");
            JCoParameterList imports=stfcConnection.getImportParameterList();
            imports.setValue("REQUTEXT", "SAP Connectivity service runs with
JCo");
            stfcConnection.execute(destination);
            JCoParameterList exports=stfcConnection.getExportParameterList();
            String echotext=exports.getString("ECHOTEXT");
            String resptext=exports.getString("RESPTEXT");
            response.addHeader("Content-type", "text/html");
            responseWriter.println("<html><body>");
            responseWriter.println("<h1>Executed STFC_CONNECTION in system
JCoDemoSystem</h1>");
            responseWriter.println("<p>Export parameter ECHOTEXT of
STFC_CONNECTION:<br>");
            responseWriter.println(echotext);
            responseWriter.println("<p>Export parameter RESPTEXT of
STFC_CONNECTION:<br>");
            responseWriter.println(resptext);
            responseWriter.println("</body></html>");
        }
        catch (AbapException ae)
        {
```

```
            //just for completeness: As this function module does not have an
exception
            //in its signature, this exception cannot occur. However,you
should always
            //take care of AbapExceptions
        }
        catch (JCoException e)
        {
            response.addHeader("Content-type", "text/html");
            responseWriter.println("<html><body>");
            responseWriter.println("<h1>Exception occurred while executing
STFC_CONNECTION in system JCoDemoSystem</h1>");
            responseWriter.println("<pre>");
            e.printStackTrace(responseWriter);
            responseWriter.println("</pre>");
            responseWriter.println("</body></html>");
        }
    }
}
```

5. Save the Java editor and make sure that the project compiles without errors.

Deploying the Application

Procedure

1. To deploy your Web application locally or on the cloud, see the following two procedures, respectively:
   -
   -
2. Once the application is successfully deployed on the cloud and you execute it, the application throws an exception. This exception says that the *JCoDemoSystem* destination has not been specified yet:

```
Exception occurred while executing STFC_CONNECTION in system JCoDemoSystem
com.sap.conn.jco.JCoException: (106) JCO_ERROR_RESOURCE: Destination
JCoDemoSystem does not exist
    at
com.sap.conn.jco.rt.DefaultDestinationManager.update(DefaultDestinationManager
.java:223)
    at
com.sap.conn.jco.rt.DefaultDestinationManager.searchDestination(DefaultDestina
tionManager.java:377)
    at
com.sap.conn.jco.rt.DefaultDestinationManager.getDestinationInstance(DefaultDe
stinationManager.java:96)
    at
com.sap.conn.jco.JCoDestinationManager.getDestination(JCoDestinationManager.ja
va:52)
    at
com.sap.demo.jco.ConnectivityRFCExample.doGet(ConnectivityRFCExample.java:47)
        ..... (cut rest of the call stack)
```

3. As a next step, you need to configure the *JCoDemoSystem* destination.

Configuring the RFC Destination on the Cloud

To configure the destination on SAP BTP, you need to use a virtual application server host name (**`abapserver.hana.cloud`**) and a virtual system number (**`42`**) that you will expose later in the Cloud Connector. Alternatively, you could use a load balancing configuration with a message server host and a system ID.

Procedure

1. Create a properties file with the following settings:

```
Name=JCoDemoSystem
Type=RFC
jco.client.ashost=abapserver.hana.cloud
jco.client.cloud_connector_version=2
jco.client.sysnr=42
jco.client.user=DEMOUSER
jco.client.passwd=<password>
jco.client.client=000
jco.client.lang=EN
jco.destination.pool_capacity=5
```

2. Upload this file to your Web application in SAP BTP. For more information, see Configure Destinations from the Console Client [page 56].
3. Call the URL that references the cloud application again in the Web browser. The application should now return a different exception:

```
Exception occurred while executing STFC_CONNECTION in system JCoDemoSystem
com.sap.conn.jco.JCoException: (102) JCO_ERROR_COMMUNICATION: Opening
connection to backend failed: Opening connection denied
    at
com.sap.conn.jco.rt.MiddlewareJavaRfc.generateJCoException(MiddlewareJavaRfc.j
ava:632)
    at com.sap.conn.jco.rt.MiddlewareJavaRfc
$JavaRfcClient.connect(MiddlewareJavaRfc.java:1307)
    at com.sap.conn.jco.rt.ClientConnection.connect(ClientConnection.java:726)
    at com.sap.conn.jco.rt.PoolingFactory.init(PoolingFactory.java:107)
    at
com.sap.conn.jco.rt.ConnectionManager.createFactory(ConnectionManager.java:
316)
    at
com.sap.conn.jco.rt.DefaultConnectionManager.createFactory(DefaultConnectionMa
nager.java:46)
    at
com.sap.conn.jco.rt.ConnectionManager.getFactory(ConnectionManager.java:290)
    at com.sap.conn.jco.rt.ConnectionManager.getClient(ConnectionManager.java:
83)
    at com.sap.conn.jco.rt.Context.getConnection(Context.java:216)
    at com.sap.conn.jco.rt.RfcDestination.execute(RfcDestination.java:1306)
    at com.sap.conn.jco.rt.RfcDestination.execute(RfcDestination.java:1278)
    at com.sap.conn.jco.rt.AbapFunction.execute(AbapFunction.java:295)
    at
com.sap.demo.jco.ConnectivityRFCExample.doGet(ConnectivityRFCExample.java:55)
        ..... (cut rest of the call stack)
```

4. This means the Cloud Connector denied opening a connection to this system. As a next step, you need to configure the system in your installed Cloud Connector.

Configuring the System Mapping in the Cloud Connector

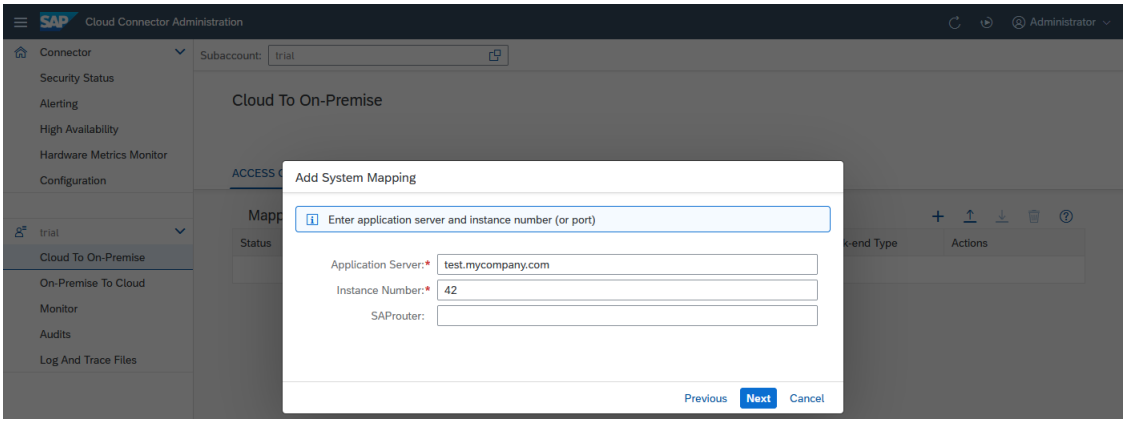
This is required since the Cloud Connector only allows access to white-listed back-end systems. To do this, follow the steps below:

Procedure

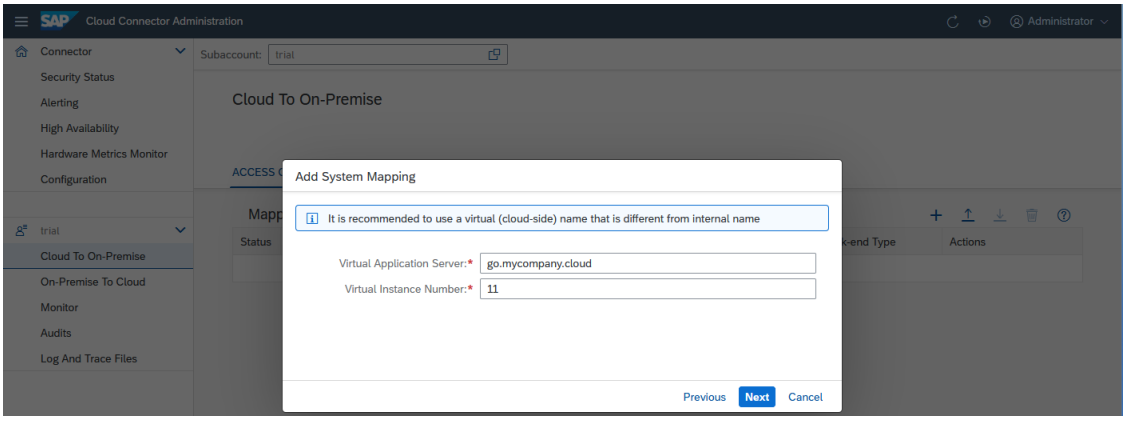
1. Optional: In the Cloud Connector administration UI, you can check under *Audits* whether access has been denied:

   ```
   Denying access for user DEMOUSER to system abapserver.hana.cloud:sapgw42
   [connectionId=-1547299395]
   ```

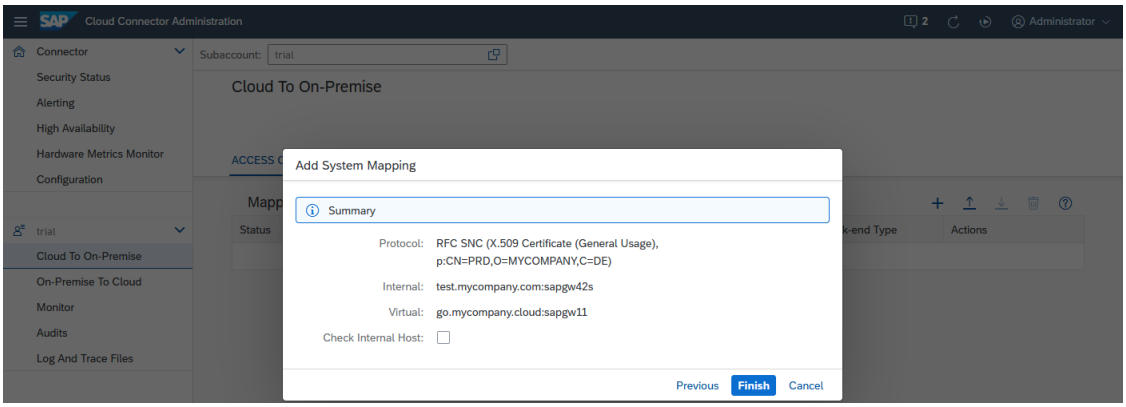
2. In the Cloud Connector administration UI and choose *Cloud To On-Premise* from your *Subaccount* menu, tab *Access Control*.
3. In section *Mapping Virtual To Internal System* choose *Add* to define a new system.
   1. For *Back-end Type*, select **`ABAP System`** and choose *Next*.
   2. For *Protocol*, select **`RFC`** and choose *Next*.
   3. Choose option *Without load balancing*.
   4. Enter application server and instance number. The *Application Server* entry must be the physical host name of the machine on which the ABAP application server is running. Choose *Next*.
      Example:

   5. Enter server and instance number for virtual mapping. **Example:**

6. Summary (example):



4. Call the URL that references the cloud application again in the Web browser. The application should now throw a different exception:

```
com.sap.conn.jco.JCoException: (102) JCO_ERROR_COMMUNICATION: Access denied
for STFC_CONNECTION
    at
com.sap.conn.jco.rt.MiddlewareJavaRfc.generateJCoException(MiddlewareJavaRfc.j
ava:632)
    at com.sap.conn.jco.rt.MiddlewareJavaRfc
$JavaRfcClient.execute(MiddlewareJavaRfc.java:1764)
    at com.sap.conn.jco.rt.ClientConnection.execute(ClientConnection.java:
1110)
    at com.sap.conn.jco.rt.ClientConnection.execute(ClientConnection.java:943)
    at com.sap.conn.jco.rt.RfcDestination.execute(RfcDestination.java:1307)
    at com.sap.conn.jco.rt.RfcDestination.execute(RfcDestination.java:1278)
    at com.sap.conn.jco.rt.AbapFunction.execute(AbapFunction.java:295)
    at
com.sap.demo.jco.ConnectivityRFCExample.doGet(ConnectivityRFCExample.java:55)
        ..... (cut rest of the call stack)
```

5. This means the Cloud Connector denied invoking STFC_CONNECTION in this system. As a final step, you need to provide access to this function module in your installed Cloud Connector.

Configuring the Function Module in the Cloud Connector

This is required since the Cloud Connector only allows access to white-listed resources (which are defined on the basis of function module names with RFC). To do this, follow the steps below:

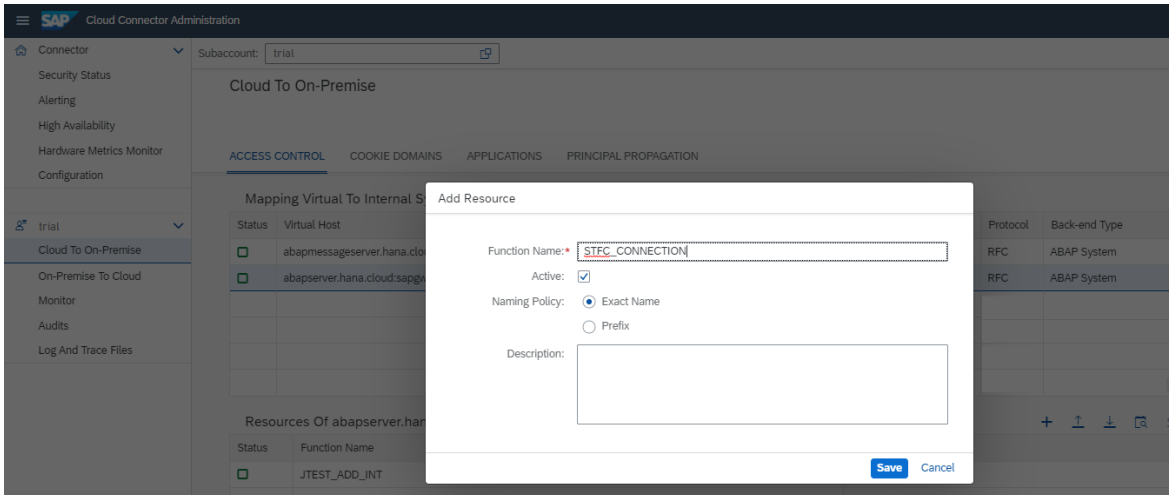
Procedure

1. Optional: In the Cloud Connector administration UI, you can check under *Audits* whether access has been denied:

```
Denying access for user DEMOUSER to resource STFC_CONNECTION on system
abapserver.hana.cloud:sapgw42 [connectionId=609399452]
```

2. In the Cloud Connector administration UI, choose *Cloud To On-Premise* from your *Subaccount* menu, and go to the *Access Control* tab.

3. For the specified internal system referring to *abapserver.hana.cloud*, add a new resource. Select the system in the table.
4. Add a new function name under the list of exposed resources. In section *Resources Accessible On abapserver.hana.cloud:sapgw42*, choose the *Add* button and specify STFC_CONNECTION as the accessible resource as shown in the screenshot below. Make sure that you have selected the *Exact Name* option to only expose this single function module.



5. Call the URL that references the cloud application again in the Web browser. The application should now return with a message showing the export parameters of the function module after a successful invocation.

Related Information

You can monitor the state and logs of your Web application deployed on SAP BTP.

For more information, see .

1.4.4.1.4 Using LDAP

Find an example how to use an LDAP destination within your cloud application.

To learn more about configuring LDAP destinations, see LDAP Destinations [page 125].

Example: LDAP Destination

Sample Code

```
package com.sap.cloud.example.ldap;

import java.io.IOException;
import java.util.Properties;

import javax.annotation.Resource;
import javax.naming.NamingEnumeration;
```

```
import javax.naming.NamingException;
import javax.naming.directory.DirContext;
import javax.naming.directory.InitialDirContext;
import javax.naming.directory.SearchControls;
import javax.naming.directory.SearchResult;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.sap.core.connectivity.api.configuration.ConnectivityConfiguration;
import com.sap.core.connectivity.api.configuration.DestinationConfiguration;

/**
 * Servlet that obtain LDAP destination, connect to the specified LDAP server
and search for users.
 */
@WebServlet("/*")
public class LdapExample extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final String DESTINATION_NAME = "example-ldap-destination";

    private static final String LDAP_PATH_TO_USERS =
"ou=users,dc=examplecompany,dc=com";
    private static final String LDAP_FILTER_MATCHING_USERS =
"(objectClass=person)";

    @Resource(name = "ConnectivityConfiguration")
    private static ConnectivityConfiguration connectivityConfiguration;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        DestinationConfiguration destination =
connectivityConfiguration.getConfiguration(DESTINATION_NAME);
        Properties properties = new Properties();
        properties.putAll(destination.getAllProperties());

        try {
            DirContext context = new InitialDirContext(properties);
            SearchControls controls = new SearchControls();
            controls.setSearchScope(SearchControls.SUBTREE_SCOPE);

            NamingEnumeration<SearchResult> result =
context.search(LDAP_PATH_TO_USERS, LDAP_FILTER_MATCHING_USERS, controls);
            response.getWriter().append("Found users:<br/><br/>");
            while (result.hasMore()) {

response.getWriter().append(result.next().toString()).append("<br/><br/>");
            }
        } catch (NamingException e) {
            throw new ServletException("Could not search LDAP for users", e);
        }
    }
}
```

1.4.4.1.5 Using the TCP Protocol for Cloud Applications

Access on-premise systems from a Neo application via TCP-based protocols, using a SOCKS5 Proxy.

Content

Concept [page 205]

Restrictions [page 205]

Example [page 206]

Interfaces [page 206]

Troubleshooting [page 207]

Concept

SAP BTP Connectivity provides a SOCKS5 proxy that you can use to access on-premise systems via TCP-based protocols. SOCKS5 is the industry standard for proxying TCP-based traffic (for more information, see IETF RFC 1928).

The proxy server is started by default on all application machines. So you can access it on `localhost` and port `20004`.

The use of SOCK5 *Username/Password Authentication* is mandatory as it is required to provide routing information to the proxy, see RFC 1929 .

In this scenario, it is used to find the correct Cloud Connector to which the data will be routed. Therefore, the pattern used for `username` is `1.<subaccount>.<locationId>`, where `subaccount` is a mandatory parameter, whereas `locationId` is optional.

> **i Note**
>
> Both values should be base64-encoded.

The Cloud Connector location ID identifies Cloud Connector instances that are deployed in various locations of a customer's premises and connected to the same subaccount. Since the location ID is an optional property, you should include it in the request only if it has already been configured in the Cloud Connector. For more information, see Set up Connection Parameters and HTTPS Proxy [page 281] (Step 4).

The `password` part of the authentication scheme is left as an empty string in this scenario.

Back to Content [page 205]

Restrictions

- You can use the provided SOCKS5 proxy server only to connect to on-premise systems. You cannot use it as general-purpose SOCKS5 proxy.
- Proxying UDP traffic is not supported.

Back to Content [page 205]

Example

The following code snippet shows how to provide the proxy authentication values :

Sample Code

```
import java.net.Authenticator;
import org.apache.commons.codec.binary.Base64; // Or any other Base64 encoder

private void setSOCKS5ProxyAuthentication(String subaccount, String
locationId){
    final String encodedSubaccount = new
String(Base64.encodeBase64(subaccount.getBytes()));
    final String encodedLocationId = new
String(Base64.encodeBase64(locationId.getBytes()));

    Authenticator.setDefault(new Authenticator() {
        @Override
        protected java.net.PasswordAuthentication getPasswordAuthentication()
{
            return new java.net.PasswordAuthentication("1." +
encodedSubaccount + "." + encodedLocationId , new char[]{});
        }
    });
}
```

In this code snippet you can see how to set up the SOCKS proxy and how to use it to create an HTTP connection:

Sample Code

```
import java.net.SocketAddress;
import java.net.InetSocketAddress;
import java.net.Proxy;
import java.net.URL;
import java.net.HttpURLConnection;
    SocketAddress addr = new InetSocketAddress("localhost", 20004);
    Proxy proxy = new Proxy(Proxy.Type.SOCKS, addr);
    setSOCKS5ProxyAuthentication(subaccount, locationId); // where subaccount
is the current subaccount and locationId is the Location ID of the SCC (or
empty string if locationId is not set)
    URL url = new URL("http://virtualhost:1234/");
    HttpURLConnection connection = (HttpURLConnection)
url.openConnection(proxy);
```

Back to Content [page 205]

Interfaces

You can access a subaccount associated with the current execution thread using the `TenantContext` API.

See also:

- Interface TenantContext
- Interface TenantContext: getTenant()
- Interface Tenant: getAccount()
- Interface Account: getId()

Back to Content [page 205]

Troubleshooting

If the handshake with the SOCKS5 proxy server fails, a SOCKS5 protocol error is returned, see IETF RFC 1928 . The table below shows the most common errors and their root cause in the scenario you use:

SOCSK5 Error Code	Technical Description	Client-Side Error Description	Scenario Error
0x00	SUCCESS		Success
0x01	FAILURE		Connection closed by backend or general scenario failure.
0x02	FORBIDDEN	`Connection not allowed by ruleset`	No matching host mapping found in Cloud Connector access control settings, see Configure Access Control (TCP) [page 347].
0x03	NETWORK_UNREACHABLE		The Cloud Connector is not connected to the subaccount and the Cloud Connector `Location ID` that is used by the cloud application can't be identified. See Connect and Disconnect a Cloud Subaccount [page 476] and Managing Subaccounts [page 289], section *Procedure*.
0x04	HOST_UNREACHABLE		Cannot open connection to the backend, that is, the host is unreachable.
0x05	CONNECTION_REFUSED		*Not used*
0x06	TTL_EXPIRED		*Not used*
0x07	COMMAND_UNSUPPORTED		Only the SOCKS5 `CONNECT` command is supported.

SOCSK5 Error Code	Technical Description	Client-Side Error Description	Scenario Error
0x08	ADDRESS_UNSUPPORTED		Only the SOCKS5 `DOMAIN` and `IPv4` commands are supported.

Back to Content [page 205]

Related Information

Configure Access Control (TCP) [page 347]

1.4.4.1.6 Sending and Fetching E-Mail

E-mail connectivity lets you send messages from your Web applications using e-mail providers that are accessible on the Internet, as well as retrieve e-mails from the mailbox of your e-mail account.

> **i Note**
>
> SAP does not act as e-mail provider. To use this service, please cooperate with an external e-mail provider of your choice.

To send and fetch e-mail, you must do the following:

- Obtain a mail session resource using resource injection or, alternatively, using a JNDI lookup.
- Configure the mail session resource by specifying the protocol settings of your mail server as a mail destination configuration. SMTP is supported for sending e-mail, and POP3 and IMAP for retrieving messages from a mailbox account.
- In your Web application, use the JavaMail API (`javax.mail`) to create and send a `MimeMessage` object or retrieve e-mails from a message store.

Related Information

Mail Destinations [page 127]
JavaMail API [page 209]
Enable the Debugging Feature [page 210]
Example: Send E-Mails [page 211]

1.4.4.1.6.1 JavaMail API

In your Web application, you use the JavaMail API (`javax.mail`) to create and send a `MimeMessage` object or retrieve e-mails from a message store.

Mail Session

You can obtain a mail session resource using resource injection or a JNDI lookup. The properties of the mail session are specified by a mail destination configuration. So that the resource is linked to this configuration, the names of the destination configuration and mail session resource must be the same.

- Resource injection
  You can directly inject the mail session resource using annotations as shown in the example below. You do not need to declare the JNDI resource reference in the `web.xml` deployment descriptor.

  ```
  @Resource(name = "mail/Session")
  private javax.mail.Session mailSession;
  ```

- JNDI lookup
  To obtain a resource of type `javax.mail.Session`, you declare a JNDI resource reference in the `web.xml` deployment descriptor in the `WebContent/WEB-INF` directory as shown below. Note that the recommended resource reference name is `Session` and the recommended subcontext is `mail` (`mail/Session`):

  ```
  <resource-ref>
      <res-ref-name>mail/Session</res-ref-name>
      <res-type>javax.mail.Session</res-type>
  </resource-ref>
  ```

  An initial JNDI context can be obtained by creating a `javax.naming.InitialContext` object. You can then consume the resource by looking up the naming environment through the `InitialContext`, as follows:

  ```
  InitialContext ctx = new InitialContext();
  Session mailSession = (Session)ctx.lookup("java:comp/env/mail/Session");
  ```

  Note that according to the Java EE Specification, the prefix `java:comp/env` should be added to the JNDI resource name (as specified in the `web.xml`) to form the lookup name.

Sending E-Mail

With the `javax.mail.Session` object you have retrieved, you can use the JavaMail API to create a `MimeMessage` object with its constituent parts (instances of `MimeMultipart` and `MimeBodyPart`). The message can then be sent using the `send` method from the `Transport` class:

```
Transport transport = mailSession.getTransport();
transport.connect();
MimeMessage mimeMessage = new MimeMessage(mailSession);
...
```

```
transport.sendMessage(mimeMessage, mimeMessage.getAllRecipients());
transport.close();
```

Fetching E-Mail

You can retrieve the e-mails from the inbox folder of your e-mail account using the `getFolder` method from the `Store` class as follows:

```
Store store = mailSession.getStore();
store.connect();
Folder folder = store.getFolder("INBOX");
folder.open(Folder.READ_ONLY);
Message[] messages = folder.getMessages();
...
folder.close(true);
store.close();
```

Fetched e-mail is not scanned for viruses. This means that e-mail retrieved from an e-mail provider using IMAP or POP3 could contain a virus that could potentially be distributed (for example, if e-mail is stored in the database or forwarded). Basic mitigation steps you could take include the following:

- Choose an e-mail provider that scans received e-mail for viruses
- Store e-mail in the document service repository before processing it. Make sure that the virus scanner provided by the document service is enabled.
- Generally don't resend e-mail that you have fetched

Related Information

Connectivity and Destination APIs [page 135]

1.4.4.1.6.2 Enable the Debugging Feature

In order to troubleshoot e-mail delivery and retrieval issues, it is useful to have debug information about the mail session established between your SAP BTP application and your e-mail provider.

Context

To include debug information in the standard trace log files written at runtime, you can use the JavaMail debugging feature and the `System.out` logger. The `System.out` logger is preconfigured with the log level INFO. You require at least INFO or a level with more detailed information.

Procedure

1. To enable the JavaMail debugging feature, add the `mail.debug` property to the mail destination configuration as shown below:

   ```
   mail.debug=true
   ```

2. To check the log level for your application, log on to the cockpit.
3. In the content area, choose *Applications* > *Java Applications*.
4. In the application list, select your application to go to the overview.
5. In the content area, choose *Monitoring* > *Logging*.
6. In the *Default Trace* section in the *Log Files* panel, choose *Configure Loggers*.
   In the *Logger Configuration* dialog box, all loggers used since the application was started are listed with the log levels that are currently applicable. Loggers are listed only if the relevant application code has been executed.
7. Enter **`system.out`** in the *Filter* field.
8. If necessary, change the log level for the `System.out` logger.

   > **i Note**
   >
   > You can check the log level of the `System.out` logger in a similar manner from the Eclipse IDE.

Related Information

1.4.4.1.6.3 Example: Send E-Mails

This example shows how you can send an e-mail from a simple Web application using an e-mail provider that is accessible on the Internet.

> **i Note**
>
> SAP does not act as e-mail provider. To use this service, please cooperate with an external e-mail provider of your choice.

Steps	Sample Application
Prerequisites [page 212]	The application is also available as a sample in the SAP BTP SDK: Sample name: `mail` Location: `<sdk>/samples` folder
1. Create a Dynamic Web Project and Servlet [page 212]	
2. Extend the Servlet [page 212]	
3. Test the Application Locally [page 214]	

Steps	Sample Application
4. Test the Application in the Cloud [page 215]	More information: Using Samples [page 850]

Prerequisites

You have installed the SAP BTP Tools and created an SAP HANA Cloud server runtime environment as described in Setting Up the Development Environment [page 845].

1. Create a Dynamic Web Project and Servlet

To develop applications for the SAP BTP, you require a dynamic Web project and servlet.

1. From the Eclipse main menu, choose *File* > *New* > *Dynamic Web Project*.
2. In the *Project name* field, enter **`mail`**.
3. In the *Target Runtime* pane, select the runtime you want to use to deploy the application. This example uses **`Java Web`**.
4. In the *Configuration* area, leave the default configuration and choose *Finish*.
5. To add a servlet to the project you have just created, select the `mail` node in the *Project Explorer* view.
6. From the Eclipse main menu, choose *File* > *New* > *Servlet*.
7. Enter the Java package **`com.sap.cloud.sample.mail`** and the class name **`MailServlet`**.
8. Choose *Finish* to generate the servlet.

2. Extend the Servlet

You add code to create a simple Web UI for composing and sending an e-mail message. The code includes the following methods:

- `doGet()`: Creates an HTML form for entering e-mail details.
- `doPost()`: Uses the mail session resource to create and send a `MimeMessage` object. It confirms that an e-mail has been sent.

1. In the *Project Explorer* view, expand the `mail/Java Resources/src/com.sap.cloud.sample.mail` node.
2. Select `MailServlet.java`, and from the context menu choose *Open With* > *Java Editor*.
3. In the opened editor, replace the entire servlet class with the following content:

```
package com.sap.cloud.sample.mail;
import java.io.IOException;
import java.io.PrintWriter;
import javax.annotation.Resource;
import javax.mail.Message.RecipientType;
import javax.mail.MessagingException;
import javax.mail.Session;
```

```
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
/**
 * Servlet implementing a mail example which shows how to use the Connectivity
                                    service APIs to send e-mail.
 * The example provides a simple UI to compose an e-mail message and send it.
The post method uses
 * the Connectivity
                                    service and the javax.mail API to send
the e-mail.
 */
public class MailServlet extends HttpServlet {
    @Resource(name = "mail/Session")
    private Session mailSession;
    private static final long serialVersionUID = 1L;
    private static final Logger LOGGER =
LoggerFactory.getLogger(MailServlet.class);
    /** {@inheritDoc} */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // Show input form to user
        response.setHeader("Content-Type", "text/html");
        PrintWriter writer = response.getWriter();
        writer.write("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
Transitional//EN\" "
                + "\"http://www.w3.org/TR/html4/loose.dtd\">");
        writer.write("<html><head><title>Mail Test</title></head><body>");
        writer.write("<form action='' method='post'>");
        writer.write("<table style='width: 100%'>");
        writer.write("<tr>");
        writer.write("<td width='100px'><label>From:</label></td>");
        writer.write("<td><input type='text' size='50' value=''
name='fromaddress'></td>");
        writer.write("</tr>");
        writer.write("<tr>");
        writer.write("<td><label>To:</label></td>");
        writer.write("<td><input type='text' size='50' value=''
name='toaddress'></td>");
        writer.write("</tr>");
        writer.write("<tr>");
        writer.write("<td><label>Subject:</label></td>");
        writer.write("<td><textarea rows='1' cols='100'
name='subjecttext'>Subject</textarea></td>");
        writer.write("</tr>");
        writer.write("<tr>");
        writer.write("<td><label>Mail:</label></td>");
        writer.write("<td><textarea rows='7' cols='100' name='mailtext'>Mail
Text</textarea></td>");
        writer.write("</tr>");
        writer.write("<tr>");
        writer.write("<tr>");
        writer.write("<td><input type='submit' value='Send Mail'></td>");
        writer.write("</tr>");
        writer.write("</table>");
        writer.write("</form>");
        writer.write("</body></html>");
    }
    /** {@inheritDoc} */
```

```
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException,
            IOException {
        Transport transport = null;
        try {
            // Parse form parameters
            String from = request.getParameter("fromaddress");
            String to = request.getParameter("toaddress");
            String subjectText = request.getParameter("subjecttext");
            String mailText = request.getParameter("mailtext");
            if (from.isEmpty() || to.isEmpty()) {
                throw new RuntimeException("Form parameters From and To may
not be empty!");
            }
            // Construct message from parameters
            MimeMessage mimeMessage = new MimeMessage(mailSession);
            InternetAddress[] fromAddress = InternetAddress.parse(from);
            InternetAddress[] toAddresses = InternetAddress.parse(to);
            mimeMessage.setFrom(fromAddress[0]);
            mimeMessage.setRecipients(RecipientType.TO, toAddresses);
            mimeMessage.setSubject(subjectText, "UTF-8");
            MimeMultipart multiPart = new MimeMultipart("alternative");
            MimeBodyPart part = new MimeBodyPart();
            part.setText(mailText, "utf-8", "plain");
            multiPart.addBodyPart(part);
            mimeMessage.setContent(multiPart);
            // Send mail
            transport = mailSession.getTransport();
            transport.connect();
            transport.sendMessage(mimeMessage,
mimeMessage.getAllRecipients());
            // Confirm mail sending
            response.getWriter().println(
                    "E-mail was sent (in local scenario stored in '<local-
server>/work/mailservice'"
                            + " - in cloud scenario using configured mail
session).");
        } catch (Exception e) {
            LOGGER.error("Mail operation failed", e);
            throw new ServletException(e);
        } finally {
            // Close transport layer
            if (transport != null) {
                try {
                    transport.close();
                } catch (MessagingException e) {
                    throw new ServletException(e);
                }
            }
        }
    }
}
```

4. Save the class.

3. Test the Application Locally

Test your code using the local file system before configuring your mail destination and testing the application in the cloud.

1. To test your application on the local server, select the servlet and choose *Run* > *Run As* > *Run on Server*.

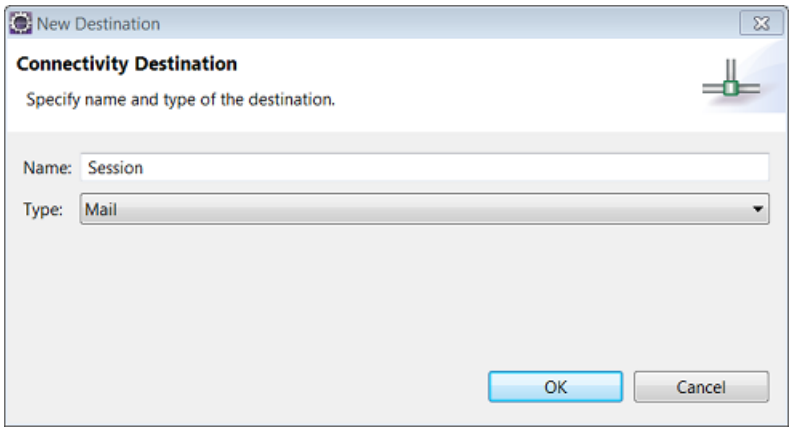
2. Make sure that the *Manually define a new server* radio button is selected and select *SAP* > *Java Web Server* .
3. Choose *Finish*. A sender screen appears, allowing you to compose and send an e-mail. The sent e-mail is stored in the `work/mailservice` directory contained in the root of your SAP BTP local runtime server.

> **i Note**
>
> To send the e-mail through a real e-mail server, you can configure a destination as described in the next section, but using the local server runtime. Remember that once you have configured a destination for local testing, messages are no longer sent to the local file system.

4. Test the Application in the Cloud

Create a mail destination that contains the SMTP settings of your e-mail provider. The name of the mail destination must match the name used in the resource reference in the `web.xml` descriptor.

1. In the Eclipse main menu, choose *File* > *New* > *Other* > *Server* > *Server* .
2. Select the server type *SAP Cloud Platform* and choose *Next*.
3. In the *SAP Cloud Platform Application* dialog box, enter the name of your application, subaccount, user, and password and choose *Finish*. The new server is listed in the *Servers* view.
4. Double-click the server and switch to the *Connectivity* tab.
5. In the *All Destinations* section, choose the *New Destination* button.
6. In the *New Destination* dialog box, enter the name **`Session`** and type **`Mail`** and choose *OK*.

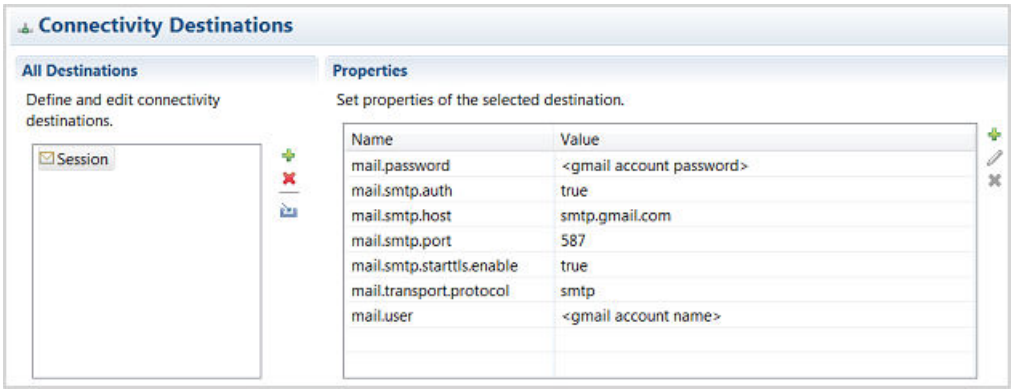


7. Configure the destination by adding the properties for port 587 (SMTP+STARTTLS) or 465 (SMTPS). To do this, choose the *Add Property* button in the *Properties* section:
   - To use port 587 (SMTP+STARTTLS), add the following properties:

Property	Value
mail.transport.protocol	smtp
mail.smtp.host	smtp.gmail.com
mail.smtp.auth	true

Property	Value
mail.smtp.starttls.enable	true
mail.smtp.port	587
mail.user	<gmail account name>
mail.password	<gmail account password>

The configured destination for port 587 is shown below:



- For port 465 (SMTPS), use the following properties:

Property	Value
mail.transport.protocol	smtps
mail.smtps.host	smtp.gmail.com
mail.smtps.auth	true
mail.smtps.port	465
mail.user	<gmail account name>
mail.password	<gmail account password>

8. Save the destination to upload it to the cloud. The settings take effect when the application is next started.
9. In the *Project Explorer* view, select `MailServlet.java` and choose *Run* > *Run As* > *Run on Server*.
10. Make sure that the *Choose an existing server* radio button is selected and select the server you have just defined.
11. Choose *Finish* to deploy to the cloud. You should now see the sender screen, where you can compose and send an e-mail

1.4.4.2 Consuming the Connectivity Service (HANA XS)

Create connectivity destinations for HANA XS applications, configure their security, add roles and test them in an enterprise or trial landscape.

- Connectivity for SAP HANA XS (Enterprise Version) [page 217]

- Connectivity for SAP HANA XS (Trial Version) [page 227]

Related Information

Multitarget Applications for the Neo Environment [page 1016]
Security

1.4.4.2.1 Connectivity for SAP HANA XS (Enterprise Version)

Overview

This section represents the usage of the Connectivity service in a productive SAP HANA instance. Below are listed the scenarios depending on the connectivity and authentication types you use for your development work.

Connectivity Types

Internet Connectivity

In this case, you can develop an XS application in a productive SAP HANA instance at SAP BTP. This enables the application to connect to external Internet services or resources.

The corresponding XS parameters for all enterprise region hosts are the same (see also Regions and Hosts Available for the Neo Environment [page 17]):

XS parameter	Value
`useProxy`	true
`proxyHost`	proxy
`proxyPort`	8080
`useSSL`	true / false

> **i** Note
>
> In the outbound scenario, the `useSSL` property can be set to **`true`** or **`false`** depending on the XS application's needs.

For more information, see Use XS Destinations for Internet Connectivity [page 219]

On-Demand to On-Premise Connectivity

In this case, you can develop an XS application in a productive SAP HANA instance at SAP BTP. That way the application connects, via a Cloud Connector tunnel, to on-premise services and resources.

The corresponding XS parameters for all enterprise regions hosts are the same (see also Regions and Hosts Available for the Neo Environment [page 17]):

XS parameter	Value
`useProxy`	true
`proxyHost`	localhost
`proxyPort`	20003
`useSSL`	false

> **i Note**
>
> When XS applications consume the Connectivity service to connect to on-premise systems, the `useSSL` property must always be set to **`false`**.
>
> The communication between the XS application and the proxy listening on localhost is always via HTTP. Whether the connection to the on-premise back end should be HTTP or HTTPS is a matter of access control configuration in the Cloud Connector. For more information, see Configure Access Control (HTTP) [page 331].

For more information, see Use XS Destinations for On-Demand to On-Premise Connectivity [page 223]

Authentication Types

No Authentication

- Internet via HTTP - you can directly connect to an Internet service.
- Internet via HTTPS - you need to use SSL certificate to access an Internet service. To meet this requirement, proceed as follows:
  1. As a prerequisite, you need to have previously exported a certificate for the relevant HTTPS site.
  2. Then, open a Web browser and start the SAP HANA XS Administration Tool (*https://<schema><account>.<host>sap/hana/xs/admin/*).
  3. On the *XS Applications* page, expand the nodes in the application tree to locate your application.
  4. Select the `.xshttpdest` file to display details of the HTTP destination and then choose *Edit*.
  5. Select the *SSL Enabled* checkbox and in the `<Trust Store>` field choose your certificate from the drop-down list.

6. As *SSL Authentication Type*, choose **`Client Certificate`**.
7. In the *Authentication Type* section, leave the *None* radio button selected.
8. Save your entries.

Basic Authentication

You need credentials to access an Internet or on-premise service. To meet this requirement, proceed as follows:

1. Open a Web browser and start the SAP HANA XS Administration Tool (*https://<schema><account>.<host>/sap/hana/xs/admin/*).
2. On the *XS Applications* page, expand the nodes in the application tree to locate your application.
3. Select the `.xshttpdest` file to display details of the HTTP destination and then choose *Edit*.
4. In the *AUTHENTICATION* section, choose the *Basic* radio button.
5. Enter the credentials for the on-premise service.
6. Save your entries.

1.4.4.2.1.1 Use XS Destinations for Internet Connectivity

Context

This section explains how to create a simple SAP HANA XS application, which is written in server-side JavaScript and makes use of theConnectivity service for making Internet connections.

In the HTTP example, the package is named **connectivity** and the XS application is **mapinfo**. The output displays information from Google Maps showing the distance between Frankfurt and Cologne, together with the consumed time if traveling with a car, as all this information is provided in American English..

> **i** Note
>
> You can check another outbound connectivity example (financial services that display the latest stock values) in *Developer Guide for SAP HANA Studio* → section "**8.4.1 Using the XSJS Outbound API** ". For more information, see the SAP HANA Developer Guides listed in the *Related Links* section below. Refer to the SAP BTP Release Notes to find out which HANA SPS is supported by SAP BTP.

Prerequisites

- You have a productive SAP HANA instance. For more information, see Using an SAP HANA XS Database System [page 999].

- You have installed the SAP HANA tools. For more information, see Install SAP HANA Tools for Eclipse [page 984].

1. Initial Steps

To create and assign an XS destination, you need to have a developed HANA XS application.

- If you have already created one and have opened a database tunnel, go straight to procedure **2. Create an XS Destination File** on this page.
- If you need to create an XS application from scratch, go to page Creating an SAP HANA XS Hello World Application Using SAP HANA Studio [page 989] and execute procedures **1** to **4**. Then execute the procedures from this page (**2** to **5**).

> **i** Note
>
> The subpackage in which you will later create your XS destination and XSJS files has to be named **connectivity**.

2. Create an XS Destination File

1. In the *Project Explorer* view, select the *connectivity* folder and choose *File* > *New* > *File*.
2. Enter the file name **`google.xshttpdest`** and choose *Finish*.
3. Copy and paste the following destination configuration settings:

```
host = "maps.googleapis.com";
port = 80;
pathPrefix = "/maps/api/distancematrix/json";
authType = none;
useSSL = false;
timeout = 30000;
```

4. Save your changes.
5. Activate the file.

3. Create an XSJS File

1. In the *Project Explorer* view, select the *connectivity* folder and choose *File* > *New* > *File*.
2. Enter the file name **`google_test.xsjs`** and choose *Finish*.
3. Copy and paste the following JavaScript code into the file:

```
var destination_package = "connectivity.mapinfo";
var destination_name = "google";
try {
       var dest = $.net.http.readDestination(destination_package,
destination_name);
       var client = new $.net.http.Client();
```

```
        var req = new $.web.WebRequest($.net.http.GET, "?
origins=Frankfurt&destinations=Cologne&mode=driving&language=en-
US&sensor=false");
        client.request(req, dest);
        var response = client.getResponse();

    $.response.contentType = "application/json";
        $.response.setBody(response.body.asString());
        $.response.status = $.net.http.OK;
} catch (e) {
        $.response.contentType = "text/plain";
        $.response.setBody(e.message);
}
```

4. Save your changes.
5. Activate the file.

> **i Note**
>
> To consume an Internet service via HTTPS, you need to export your HTTPS service certificate into `X.509` format, to import it into a trust store and to assign it to your activated destination. You need to do this in the SAP HANA XS Administration Tool (*https://<schema><account>.<host>/sap/hana/xs/admin/*). For more information, see *Developer Guide for SAP HANA Studio* → section "3.6.2.1 SAP HANA XS Application Authentication". For more information, see the SAP HANA Developer Guides listed in the *Related Links* section below. Refer to the SAP BTP Release Notes to find out which HANA SPS is supported by SAP BTP.

4. Grant a Role to the User

1. In the *Systems* view, expand *Security* > *Users* and then double-click your user ID.
2. On the *Granted Roles* tab, choose the *+ (Add)* button.
3. Select the **`model_access`** role in the list and choose *OK*. The role is now listed on the *Granted Roles* tab.
4. Choose *Deploy* in the upper right corner of screen. A message confirms that your user has been modified.

5. Test Your Application

Open the cockpit and proceed as described in Launch SAP HANA XS Applications [page 998].

You will be authenticated by SAML and should then see the following response:

```
{
   "destination_addresses" : [ "Cologne, Germany" ],
   "origin_addresses" : [ "Frankfurt, Germany" ],
   "rows" : [
      {
         "elements" : [
            {
               "distance" : {
                  "text" : "190 km",
                  "value" : 190173
               },
               "duration" : {
```

```
                    "text" : "1 hour 58 mins",
                    "value" : 7103
                },
                "status" : "OK"
            }
        ]
    }
  ],
  "status" : "OK"
}
```

Additional Example

You can also see an example for enabling server-side JavaScript applications to use the outbound connectivity API. For more information, see *Developer Guide for SAP HANA Studio* → section "8.4.1 Using the XSJS Outbound API".

> **i** Note
>
> For more information, see the SAP HANA Developer Guides listed below. Refer to the SAP BTP Release Notes to find out which HANA SPS is supported by SAP BTP.

See Also

- Introduction to SAP HANA Development
- SAP HANA Web-Based Development Workbench

Related Information

XS Destination Properties [page 226]
Consume Internet Services (Java Web or Java EE 6 Web Profile) [page 155]
Creating an SAP HANA XS Hello World Application Using SAP HANA Web-based Development Workbench [page 985]

1.4.4.2.1.2 Use XS Destinations for On-Demand to On-Premise Connectivity

Context

This section explains how to create a simple SAP HANA XS application that consumes a sample back-end system exposed via the Cloud Connector.

In this example, the XS application consumes an on-premise system with basic authentication on landscape *hana.ondemand.com*.

Prerequisites

- You have a productive SAP HANA instance. For more information, see Using an SAP HANA XS Database System [page 999].
- You have installed the SAP HANA tools. For more information, see Install SAP HANA Tools for Eclipse [page 984]. You need them to open a Database Tunnel.
- You have Cloud Connector 2.x installed on an on-premise system. For more information, see Installation [page 238].
- A sample back-end system with basic authentication is available on an on-premise host. For more information, see Set Up an Application as a Sample Backend System [page 190].
- You have created a tunnel between your subaccount and a Cloud Connector. For more information, see Initial Configuration [page 278] → section "Establishing Connections to SAP BTP".
- The back-end system is exposed for the SAP HANA XS application via Cloud Connector configuration using as settings: `virtual_host` = **virtualpingbackend** and `virtual_port` = **1234**. For more information, see Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170].

> **i** Note
>
> The last two prerequisites can be achieved by exposing any other available HTTP service in your on-premise network. In this case, you shall adjust accordingly the `pathPrefix` value, mentioned below in procedure "2. Create an XS Destination File".

1. Initial Steps

To create and assign an XS destination, you need to have a developed HANA XS application.

- If you have already created one and have opened a database tunnel, go straight to procedure **2. Create an XS Destination File** on this page.

- If you need to create an XS application from scratch, go to page Creating an SAP HANA XS Hello World Application Using SAP HANA Studio [page 989] and execute procedures **1** to **4**. Then execute the procedures from this page (**2** to **5**).

> **i Note**
>
> The subpackage in which you will later create your XS destination and XSJS files has to be named **connectivity**.

2. Create an XS Destination File

1. In the *Project Explorer* view, select the *connectivity* folder and choose *File* > *New* > *File*.
2. Enter the file name **`odop.xshttpdest`** and choose *Finish*.
3. Copy and paste the following destination configuration settings:

```
host = "virtualpingbackend";
port = 1234;
useSSL = false;
pathPrefix = "/BackendAppHttpBasicAuth/basic";
useProxy = true;
proxyHost = "localhost";
proxyPort = 20003;
timeout = 3000;
```

> **i Note**
>
> In case you use SDK with a version equal to or lower than **`1.44.0.1`** (Java Web) and **`2.24.13`** (Java EE 6 Web Profile) respectively, you should find the on-premise WAR files in directory `<SDK_location>/tools/samples/connectivity/onpremise`. Also, the pathPrefix should be `/PingAppHttpBasicAuth/pingbasic`.

4. Save your changes.
5. Activate the file.

3. Create an XSJS File

1. In the *Project Explorer* view, select the *connectivity* folder and choose *File* > *New* > *File*.
2. Enter the file name **`ODOPTest.xsjs`** and choose *Finish*.
3. Copy and paste the following JavaScript code into the file:

```
$.response.contentType = "text/html";
var dest = $.net.http.readDestination("connectivity","odop");
var client = new $.net.http.Client();
var req = new $.web.WebRequest($.net.http.GET, "");
client.request(req, dest);
var response = client.getResponse().body.asString();
$.response.setBody(response);
```

4. Save your changes.

5. Activate the file.

> **i Note**
>
> You also need to enter your on-premise credentials. You should not enter them in the destination file since they must not be exposed as plain text.

4. Provide Secured Credentials

1. Open a Web browser and start the SAP HANA XS Administration Tool (*https://<schema><account>.<host>/sap/hana/xs/admin/*).
2. On the *XS Applications* page, expand the nodes in the application tree to locate your application.
3. Select the `odop.xshttpdest` file to display the HTTP destination details and then choose *Edit*.
4. In section *AUTHENTICATION*, choose the *Basic* radio button.
5. Enter your on-premise credentials (user and password).
6. Save your entries.

> **i Note**
>
> If you later need to make another configuration change to your XS destination, you need to enter your password again since it is no longer remembered by the editor.

5. Grant a Role to the User

1. In the *Systems* view, expand *Security* > *Users* and then double-click your user ID.
2. On the *Granted Roles* tab, choose the *+ (Add)* button.
3. Select the **`model_access`** role in the list and choose *OK*. The role is now listed on the *Granted Roles* tab.
4. Choose *Deploy* in the upper right corner of screen. A message confirms that your user has been modified.

6. Test Your Application

Open the cockpit and proceed as described in Launch SAP HANA XS Applications [page 998].

Principal Propagation to on-premise systems

Principal Propagation scenario is available for HANA XS applications. It is used for propagating the currently logged in user to an on-premise back-end system using the Cloud Connector and connectivity service. To configure the scenario make sure to:

1.Create an XS destination with the parameter `authType=SamlAssertionPropagation`.

2.Open the Cloud Connector and mark your HANA instance as trusted in the *Principal Propagation* tab. The HANA instance name is displayed in the cockpit under *SAP HANA/SAP ASE* > *Databases & Schemas*. For more information, see Set Up Trust for Principal Propagation [page 301].

Related Information

XS Destination Properties [page 226]
Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170]
Creating an SAP HANA XS Hello World Application Using SAP HANA Web-based Development Workbench [page 985]

1.4.4.2.1.3 XS Destination Properties

XS Property	Description	Value
`host`	It enables you to specify the host name of the HTTP destination providing the service or data you want your SAP HANA XS application to access.	URL (string)
`port`	It enables you to specify the port number to use for connections to the HTTP destination hosting the service or data you want your SAP HANA XS application to access.	• For Internet connection: **`80`**, **`443`** • For on-demand to on-premise connection: **`1080`** • For service-to-service connection: **`8443`**
`pathPrefix`	It enables you to specify a text element to add to the start of the URL used for connections to the service specified in the HTTP destination configuration.	`"..."`; (string)

`useProxy`	It enables you to specify whether a proxy server must be used to resolve the host name specified in the HTTP destination configuration file.
	Depends on the authentication type and the landscape on which you deploy your SAP HANA XS application.
		• For all productive landscapes:
		  ○ Internet: **`true`**
		  ○ On-premise: **`true`**
		  ○ Service-to-service: **`false`**
		i Note
		See also: Connectivity for SAP HANA XS (Enterprise Version) [page 217]

`proxyHost`	Depending on the authentication type and the landscape on which you deploy your SAP HANA XS application.	• For all productive landscapes:
		  ○ Internet: **`proxy`**
		  ○ On-premise: **`localhost`**

XS Property	Description	Value
		i Note See also: Connectivity for SAP HANA XS (Enterprise Version) [page 217]
`proxyPort`	Depending on the authentication type and the landscape on which you deploy your SAP HANA XS application.	• For all productive landscapes: ○ Internet: **`8080`** ○ On-premise: **`20003`** ○ Service-to-service: **`8080`** **i** Note See also: Connectivity for SAP HANA XS (Enterprise Version) [page 217]
`authType`	It enables you to specify the authentication method that must be used for connection requests for the service located at the HTTP destination specified in the configuration.	**`none`**, **`basic`**, **`AssertionTicket`**, **`SAML Assertion Propagation`**
`useSSL`	It is of type Boolean and enables you to specify whether the outbound connection between SAP HANA XS and the HTTP destination is secured with the Secure Sockets Layer (SSL) protocol (HTTPS).	**`true`**, **`false`** **i** Note See also: Connectivity for SAP HANA XS (Enterprise Version) [page 217]
`timeout`	It enables you to specify for how long (in milliseconds) an application tries to connect to the remote host specified in the HTTP destination configuration.	**`-1`** (default) In the connectivity examples: **`3000`**

Related Information

- HTTP Destination Configuration Syntax
- Regions

1.4.4.2.2 Connectivity for SAP HANA XS (Trial Version)

Context

This section represents the usage of the Connectivity service when you develop and deploy SAP HANA XS applications in a trial environment. Currently, you can make XS destinations for consuming HTTP Internet services only.

The procedure below lets you create a simple SAP HANA XS application which is written in server-side JavaScript and makes use of the Connectivity service for making Internet connections. In the HTTP example, the package is named **connectivity** and the XS application is **mapinfo**. The output displays information from Google Maps showing the distance between Frankfurt and Cologne, together with the consumed time if traveling with a car, as all this information is provided in American English.

Features

In this case, you can develop an XS application in a trial environment at SAP BTP so that the application connects to external Internet services or resources.

XS parameter	hanatrial.ondemand.com
`useProxy`	true
`proxyHost`	proxy-trial
`proxyPort`	8080
`useSSL`	true / false

> **i** Note
>
> The `useSSL` property can be set to **`true`** or **`false`** depending on the XS application's needs.

1. Initial Steps

To create and assign an XS destination, you need to have a developed HANA XS application.

- If you have already created one and have opened a database tunnel, go straight to procedure **2. Create an XS Destination File** on this page.
- If you need to create an XS application from scratch, go to page Creating an SAP HANA XS Hello World Application Using SAP HANA Studio [page 989] and execute procedures **1** to **4**. Then execute the procedures from this page (**2** to **5**).

> **i** Note
>
> The subpackage in which you will later create your XS destination and XSJS files has to be named **connectivity**.

2. Create an XS Destination File

1. In the *Project Explorer* view, select the *connectivity* folder and choose *File* > *New* > *File*.
2. Enter the file name **`google.xshttpdest`** and choose *Finish*.
3. Copy and paste the following destination configuration settings:

```
host = "maps.googleapis.com";
port = 80;
pathPrefix = "/maps/api/distancematrix/json";
useProxy = true;
proxyHost = "proxy-trial";
proxyPort = 8080;
authType = none;
useSSL = false;
timeout = 30000;
```

4. Save your changes.
5. Activate the file.

3. Create an XSJS File

1. In the *Project Explorer* view, select the *connectivity* folder and choose *File* > *New* > *File*.
2. Enter the file name **`google_test.xsjs`** and choose *Finish*.
3. Copy and paste the following JavaScript code into the file:

```
var destination_package = "connectivity.mapinfo";
var destination_name = "google";
try {
       var dest = $.net.http.readDestination(destination_package,
destination_name);
       var client = new $.net.http.Client();
       var req = new $.web.WebRequest($.net.http.GET, "?
origins=Frankfurt&destinations=Cologne&mode=driving&language=en-
US&sensor=false");
       client.request(req, dest);
       var response = client.getResponse();

    $.response.contentType = "application/json";
       $.response.setBody(response.body.asString());
       $.response.status = $.net.http.OK;
} catch (e) {
       $.response.contentType = "text/plain";
       $.response.setBody(e.message);
}
```

4. Save your changes.
5. Activate the file.

4. Grant a Role to the User

1. In the *Systems* view, select your system and from the context menu choose *SQL Console*.

2. In the SQL console, enter the following, replacing `<SAP HANA Cloud user>` with your user:

```
call
"HCP"."HCP_GRANT_ROLE_TO_USER"('p1234567890trial.myhanaxs.hello::model_access'
, '<SAP HANA Cloud user>')
```

3. Execute the procedure. You should see a confirmation that the statement was successfully executed.

5. Test Your Application

Open the cockpit and proceed as described in Launch SAP HANA XS Applications [page 998].

You will be authenticated by SAML and should then see the following response:

```
{
   "destination_addresses" : [ "Cologne, Germany" ],
   "origin_addresses" : [ "Frankfurt, Germany" ],
   "rows" : [
      {
         "elements" : [
            {
               "distance" : {
                  "text" : "190 km",
                  "value" : 190173
               },
               "duration" : {
                  "text" : "1 hour 58 mins",
                  "value" : 7103
               },
               "status" : "OK"
            }
         ]
      }
   ],
   "status" : "OK"
}
```

Related Information

Creating an SAP HANA XS Hello World Application Using SAP HANA Web-based Development Workbench [page 985]

1.4.4.3 Consuming the Destination Configuration Service

Retrieve destination configurations for your cloud application in the Neo environment, in a secure and reliable way.

Using the Destination Configuration service, you can create, edit, update and read destinations, keystores and certificates on application, subaccount, or subscription level, see Managing Destinations [page 54]. You can access these destinations through your application at runtime or from the SAP BTP cockpit, see Configure Destinations from the Cockpit [page 77].

Prerequisites

You must have administrative access to a subaccount within the Neo environment.

Required Credentials

The Destination Configuration service requires OAuth 2.0 credentials for all REST API methods. To manage and read destinations and certificates, you must create an OAuth client and assign any of the following permissions: *Manage Destinations* (read/write) or *Read Destination* (read only), see Using Platform APIs [page 1149].

Destination Configuration Service REST API

The Destination Configuration service provides a REST API, which lets you configure the destinations that you need to connect your application to another system or service, see SAP API Business Hub.

For the `create` and `update` methods, you must send the destination values as a `properties` file with the `multipart/form-data` media type as form data.

> **Note**
>
> When you read a destination, for security reasons the same file is downloaded without the sensitive properties.

Example: Create and Read Destination Configuration Requests on Tenant Level

> **Sample Code**

```
curl -X POST https://api.{landscape_domain}.configuration/api/rest/oauth/
SPACES/{account}/connectivity/ -H 'authorization: Bearer {access_token}' -H
'content-type: multipart/form-data' -F
'{configuration_name}=@{configuration_file_path}' -v
curl -O GET https://api.{landscape_domain}.configuration/api/rest/oauth/
SPACES/{account}/connectivity/{configuration_name} -H 'authorization: Bearer
{access_token}' -H 'content-type: application/octet-stream'
```

Destination Configuration Properties

Depending on the authentication type used, different properties are required for a destination. Find the available properties for each authentication type in HTTP Destinations [page 91].

1.4.5 Cloud Connector (Neo Environment)

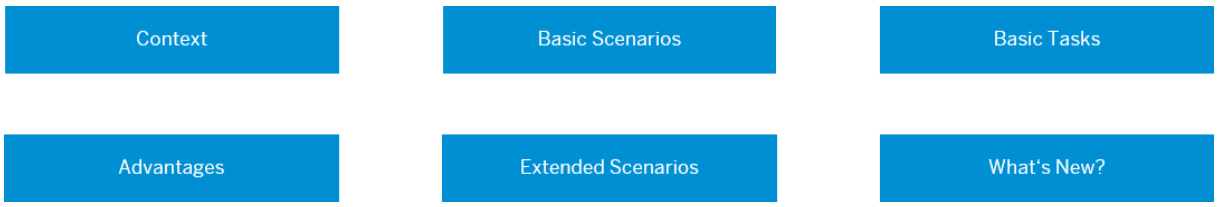
Learn more about the Cloud Connector: features, scenarios and setup.

> **i Note**
>
> This documentation refers to SAP BTP, Neo environment. If you are looking for information about the Cloud Foundry environment, see Connectivity (Cloud Foundry environment).

Content

In this Topic

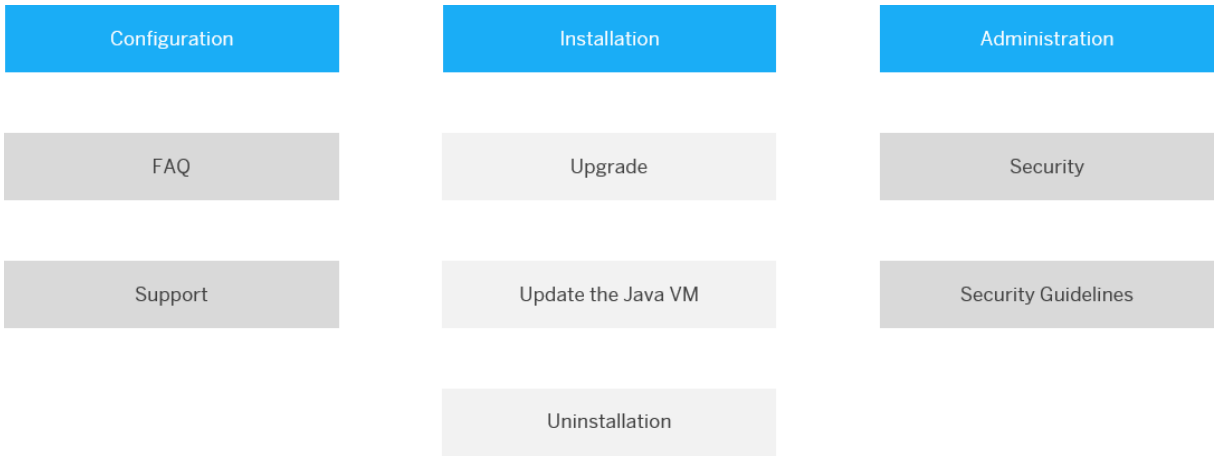
Hover over the elements for a description. Click an element for more information.



- Context [page 233]
- Basic Scenarios [page 234]
- Basic Tasks [page 237]
- Advantages [page 234]
- Extended Scenarios [page 236]
- What's New? [page 237]

In this Guide

Hover over the elements for a description. Click an element for more information.



- Configuration [page 277]
- Installation [page 238]
- Administration [page 464]
- Frequently Asked Questions [page 535]
- Upgrade [page 530]
- Security [page 519]
- Update the Java VM [page 532]
- Uninstallation [page 534]
- Connectivity Support [page 544]
- Security Guidelines [page 526]

Context

The Cloud Connector:

- Serves as a link between SAP BTP applications and on-premise systems.
  - Combines an easy setup with a clear configuration of the systems that are exposed to the SAP BTP.
  - Lets you use existing on-premise assets without exposing the entire internal landscape.
- Runs as on-premise agent in a secured network.
  - Acts as a reverse invoke proxy between the on-premise network and SAP BTP.
- Provides fine-grained control over:
  - On-premise systems and resources that can be accessed by cloud applications.
  - Cloud applications using the Cloud Connector.
- Lets you use the features that are required for business-critical enterprise scenarios.
  - Recovers broken connections automatically.
  - Provides audit logging of inbound traffic and configuration changes.
  - Can be run in a high-availability setup.

> ⚠ Caution
>
> The Cloud Connector must not be used to connect to products other than SAP BTP or S/4HANA Cloud.

Back to Content [page 232]

Advantages

Compared to the approach of opening ports in the firewall and using reverse proxies in the DMZ to establish access to on-premise systems, the Cloud Connector offers the following benefits:

- You don't need to configure the on-premise firewall to allow external access from SAP BTP to internal systems. For allowed outbound connections, no modifications are required.
- The Cloud Connector supports HTTP as well as additional protocols. For example, the RFC protocol supports native access to ABAP systems by invoking function modules.
- You can use the Cloud Connector to connect on-premise databases or BI tools to SAP HANA databases in the cloud.
- The Cloud Connector lets you propagate the identity of cloud users to on-premise systems in a secure way.
- Easy installation and configuration, which means that the Cloud Connector comes with a low TCO and is tailored to fit your cloud scenarios.
- SAP provides standard support for the Cloud Connector.

Back to Content [page 232]

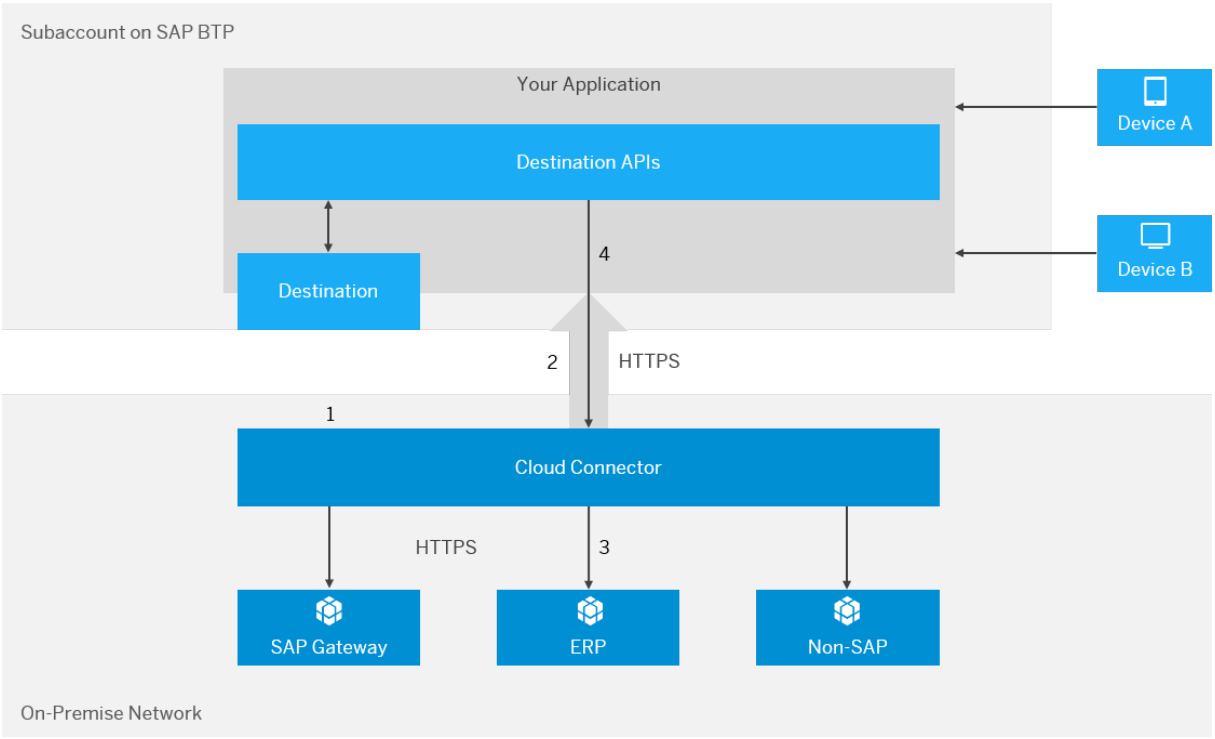
Basic Scenarios

Connecting Cloud Applications to On-Premise Systems [page 234]

Connecting On-Premise Database Tools to SAP HANA Databases [page 235]

> ℹ Note
>
> This section refers to the Cloud Connector installation in a standard on-premise network. Find setup options for other system environments in Extended Scenarios [page 236].

Connecting Cloud Applications to On-Premise Systems

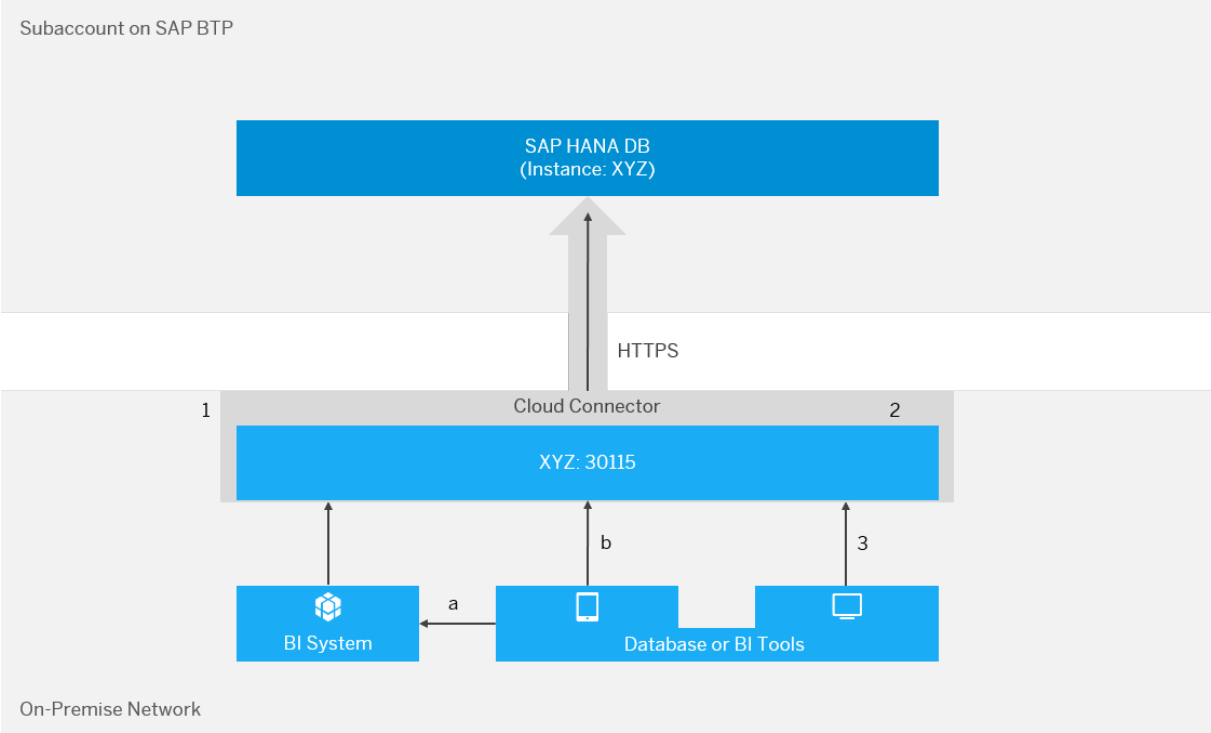


1. Install the Cloud Connector: Installation [page 238]
2. Set up the connection between Cloud Connector, back-end system and your SAP BTP subaccount : Initial Configuration [page 278], Managing Subaccounts [page 289]
3. Allow your cloud application to access a back-end system on the intranet: Configure Access Control [page 330]
4. Connect your cloud application to an on-premise system:
   Consuming the Connectivity Service [page 134] (Neo environment)

Back to Basic Scenarios [page 234]

Back to Content [page 232]

Connecting On-Premise Database Tools to SAP HANA Databases



1. Install and configure the Cloud Connector: Installation [page 238], Initial Configuration [page 278], Managing Subaccounts [page 289]
2. Access HANA databases on SAP BTP: Configure a Service Channel for an SAP HANA Database [page 444]
3. Connect on-premise database or BI tools to a HANA database on SAP BTP: Connect DB Tools to SAP HANA via Service Channels [page 447]

> **i Note**
>
> You can use service channels also for other purposes:
>
> - Connect to a virtual machine on SAP BTP.
> - Configure an RFC connection from your on-premise system to S/4HANA Cloud.
>
> See Using Service Channels [page 443].

Back to Basic Scenarios [page 234]

Back to Content [page 232]

Extended Scenarios

Besides the standard setup: *SAP BTP - Cloud Connector - on-premise system/network*, you can also use the Cloud Connector to connect SAP BTP applications to other cloud-based environments, as long as they are operated in a way that is comparable to an on-premise network from a functional perspective. This is particularly true for infrastructure (IaaS) hosting solutions.

Here's an overview of all environments in which you can or cannot set up the Cloud Connector:

Cloud Connector	Environment	Examples
Can be set up in: **i Note** Within extended scenarios that allow a Cloud Connector setup, special procedures may apply for configuration. If so, they are mentioned in the corresponding configuration steps.	Customer on-premise network (see Basic Scenarios [page 234])	SAP ERP, SAP S/4HANA
	SAP Hosting	SAP HANA Enterprise Cloud (HEC)
	Third-party **IaaS** providers (hosting)	Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP)
Cannot be set up in:	SAP **SaaS** solutions	SAP SuccessFactors, SAP Concur, SAP Ariba
	SAP cloud-based enterprise solutions	SAP S/4HANA Cloud, SAP C/4HANA
	Third-party **PaaS** providers	AWS, Azure, GCP
	Third-party **SaaS** providers	

Back to Content [page 232]

Basic Tasks

The following steps are required to connect the Cloud Connector to your SAP BTP subaccount:

- Install the Cloud Connector: Installation [page 238]
- Perform the initial configuration for the Cloud Connector: Initial Configuration [page 278]
- Register the Cloud Connector for your SAP BTP subaccount: Managing Subaccounts [page 289]

Back to Content [page 232]

What's New?

Follow the SAP BTP Release Notes to stay informed about Cloud Connector and Connectivity updates.

Back to Content [page 232]

Related Information

Installation [page 238]
Configuration [page 277]
Administration [page 464]
Security [page 519]
Upgrade [page 530]
Update the Java VM [page 532]
Uninstallation [page 534]
Frequently Asked Questions [page 535]

1.4.5.1 Installation

Choose a procedure to install the Cloud Connector on your operating system.

Portable Version vs. Installer Version

On **Microsoft Windows** and **Linux**, two installation modes are available: a `portable` version and an `installer` version. On **Mac OS X**, only the `portable` version is available.

- `Portable` version: can be installed easily, by extracting a compressed archive into an empty directory. It does not require administrator or root privileges for the installation, and you can run multiple instances on the same host.
  Restrictions:
  - You cannot run it in the background as a Windows Service or Linux daemon (with automatic start capabilities at boot time).
  - The portable version does not support an automatic upgrade procedure. To update a portable installation, you must delete the current one, extract the new version, and then re-do the configuration.
  - Portable versions are meant for non-productive scenarios only.
  - The environment variable `JAVA_HOME` is relevant when starting the instance, and therefore must be set properly.
- `Installer` version: requires administrator or root permissions for the installation and can be set up to run as a Windows service or Linux daemon in the background. You can upgrade it easily, retaining all the configuration and customizing.

  > **i Note**
  >
  > We strongly recommend that you use this variant for a productive setup.

> **⚠ Caution**
>
> Cloud Connector is based on Tomcat.
>
> Tomcat is a well-known and well-documented server, whose configuration can be adapted to one's own needs. However, we strongly recommend that you **do not modify** its configuration files like `conf`

`\server.xml` with a text editor, because the Cloud Connector is making assumptions about the content of those files.

If you still want to do custom modifications, you do so at your own risk. In this case, we can no longer guarantee that the Cloud Connector keeps working as expected. For any issues that can be traced back to such changes, we cannot provide support, in particular, if such changes cause trouble during upgrade to a newer version.

Prerequisites

- There are some general prerequisites you must fulfill to successfully install the Cloud Connector, see Prerequisites [page 239].
- For OS-specific requirements and procedures, see section **Tasks** below.

Tasks

- Installation on Microsoft Windows OS [page 259]
- Installation on Linux OS [page 262]
- Installation on Mac OS X [page 265]

Related Information

Sizing Recommendations [page 252]
Recommendations for Secure Setup [page 266]
[Deprecated] Replace the Default SSL Certificate [page 272]
Uninstallation [page 534]

1.4.5.1.1 Prerequisites

Prerequisites for successful installation of the Cloud Connector.

Content

Section	Description
Connectivity Restrictions [page 240]	General information about SAP BTP and connectivity restrictions.
Hardware [page 240]	Hardware prerequisites for a physical or virtual machine.
Software [page 241]	Required software download and installation.
JDKs [page 241]	Java Development Kit (JDK) versions that you can use.
Product Availability Matrix [page 241]	Availability of operating systems/versions for specific Cloud Connector versions.
Network [page 242]	Required Internet connection to SAP BTP hosts per region.

> **i Note**
>
> For additional system requirements, see also System Requirements [page 251].

Connectivity Restrictions

For specific information about all Connectivity restrictions, see Connectivity for the Neo Environment: Restrictions [page 24].

Back to Content [page 239]

Hardware

Hardware prerequisites, physical or virtual machine:

	Minimum	Recommended
CPU	Single core 3 GHz, x86-64 architecture compatible	Dual core 2 GHz, x86-64 architecture compatible
Memory (RAM)	2 GB	4 GB
Free disk space	3 GB	20 GB

Back to Content [page 239]

Software

- You have downloaded the Cloud Connector installation archive from SAP Development Tools for Eclipse.
- A JDK 8 must be installed. You can download an up-to-date SAP JVM from SAP Development Tools for Eclipse as well.

> ⚠ Caution
>
> Do not use Apache Portable Runtime (APR) on the system on which you use the Cloud Connector. If you cannot avoid this restriction and want to use APR at your own risk, you must manually adopt the `server.xml` configuration file in directory `<scc_installation_folder>/conf`. To do so, follow the steps in HTTPS port configuration for APR.

Back to Content [page 239]

JDKs

JDK	Version	Cloud Connector Version
SAP JVM 64-bit (recommended)	7	2.x up to 2.12.2
	8	2.7.2 and higher
Oracle JDK 64-bit	7	2.x up to 2.12.2
	8	2.7.2 and higher

> i Note
>
> The support for using Cloud Connector with Java runtime version 7 ended on December 31, 2019. Any Cloud Connector version released after that date may contain Java byte code requiring at least a JVM 8.
>
> We therefore strongly recommend that you perform **fresh** installations only with **Java 8**, and **update** existing installations running with Java 7, to Java 8.
>
> See SAP Cloud Connector – Java 7 support will phase out and Update the Java VM [page 532].

Back to Content [page 239]

Product Availability Matrix

Operating System Version	Architecture	Cloud Connector Version
Windows 7, Windows Server 2008 R2	x86_64	2.x

Operating System Version	Architecture	Cloud Connector Version
SUSE Linux Enterprise Server 11, Redhat Enterprise Linux 6	x86_64	2.x
Mac OS X 10.7 (Lion), Mac OS X 10.8 (Mountain Lion)	x86_64	2.x
Windows 8.1, Windows Server 2012, Windows Server 2012 R2	x86_64	2.5.1 and higher
SUSE Linux Enterprise Server 12, Redhat Enterprise Linux 7	x86_64	2.5.1 and higher
Mac OS X 10.9 (Mavericks), Mac OS X 10.10 (Yosemite)	x86_64	2.5.1 and higher
Windows 10	x86_64	2.7.2 and higher
Mac OS X 10.11 (El Capitan)	x86_64	2.8.1 and higher
Windows Server 2016	x86_64	2.9.1 and higher
Windows Server 2019, Mac OS X 10.12 (Sierra), Mac OS X 10.13 (High Sierra), Mac OS X 10.14 (Mojave)	x86_64	2.11.3 and higher
SUSE Linux Enterprise Server 15	x86_64	2.12.0 and higher
Redhat Enterprise Linux 8	x86_64	2.12.2 and higher
SUSE Linux Enterprise Server 12, SUSE Linux Enterprise Server 15, Redhat Enterprise Linux 7, Redhat Enterprise Linux 8	ppc64le	2.13.0 and higher

Back to Content [page 239]

Network

You must have Internet connection at least to the following Connectivity service hosts (depending on the region), to which you can connect your Cloud Connector. All connections to the hosts are TLS-based and connect to port 443.

> → Remember
>
> For some solutions of the BTP portfolio, you must include additional hosts to set up an on-premise connectivity scenario with the Cloud Connector. This applies, for example, to: SAP Data Intelligence, SAP HANA Cloud, and Business Appilcation Studio. Check the respective solution documentation for details.

> i Note
>
> For general information on IP ranges per region, see Regions (Cloud Foundry and ABAP environment) or Regions and Hosts Available for the Neo Environment. Find detailed information about the region status and planned network updates on http://sapcp.statuspage.io/ .

Cloud Foundry Environment [page 243]

ABAP Environment [page 246]

Neo Environment [page 246]

Trial [page 248]

Region (Region Host)	Hosts	IP Address
Cloud Foundry Environment		
i Note In the Cloud Foundry environment, IPs are controlled by the respective IaaS provider - Amazon Web Services (AWS), Microsoft Azure (Azure), or Google Cloud Platform (GCP). IPs may change due to network updates on the provider side. Any planned changes will be announced at least 4 weeks before they take effect. See also Regions.		
Europe (Frankfurt) - AWS (**cf.eu10.hana.ondemand.com**) *Enterprise & Trial*	connectivitynotification.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223
	connectivitycertsigning.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223
	connectivitytunnel.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223
Europe (Frankfurt) - AWS (**cf.eu11.hana.ondemand.com**)	connectivitynotification.cf.eu11.hana.ondemand.com	3.124.207.41, 18.157.105.117, 18.156.209.198
	connectivitycertsigning.cf.eu11.hana.ondemand.com	3.124.207.41, 18.157.105.117, 18.156.209.198
	connectivitytunnel.cf.eu11.hana.ondemand.com	3.124.207.41, 18.157.105.117, 18.156.209.198
Europe (Netherleands) - Azure (**cf.eu20.hana.ondemand.com**)	connectivitynotification.cf.eu20.hana.ondemand.com	40.119.153.88
	connectivitycertsigning.cf.eu20.hana.ondemand.com	40.119.153.88
	connectivitytunnel.cf.eu20.hana.ondemand.com	40.119.153.88
US East (VA) - AWS (**cf.us10.hana.ondemand.com**) *Enterprise & Trial*	connectivitynotification.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211
	connectivitycertsigning.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211

Region (Region Host)	Hosts	IP Address
	connectivitytunnel.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211
US West (WA) - Azure (**cf.us20.hana.ondemand.com**)	connectivitynotification.cf.us20.hana.ondemand.com	40.91.120.100
	connectivitycertsigning.cf.us20.hana.ondemand.com	40.91.120.100
	connectivitytunnel.cf.us20.hana.ondemand.com	40.91.120.100
US East (VA) - Azure (**cf.us21.hana.ondemand.com**)	connectivitynotification.cf.us21.hana.ondemand.com	40.88.52.17
	connectivitycertsigning.cf.us21.hana.ondemand.com	40.88.52.17
	connectivitytunnel.cf.us21.hana.ondemand.com	40.88.52.17
US Central (IA) - GCP (**cf.us30.hana.ondemand.com**)	connectivitynotification.cf.us30.hana.ondemand.com	35.184.169.79
	connectivitycertsigning.cf.us30.hana.ondemand.com	35.184.169.79
	connectivitytunnel.cf.us30.hana.ondemand.com	35.184.169.79
Brazil (São Paulo) - AWS (**cf.br10.hana.ondemand.com**)	connectivitynotification.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4
	connectivitycertsigning.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4
	connectivitytunnel.cf.br10.hana.ondemand.com	18.229.91.150, 52.67.135.4
Japan (Tokyo) - AWS (**cf.jp10.hana.ondemand.com**)	connectivitynotification.cf.jp10.hana.ondemand.com	13.114.117.83, 3.114.248.68, 3.113.252.15
	connectivitycertsigning.cf.jp10.hana.ondemand.com	13.114.117.83, 3.114.248.68, 3.113.252.15
	connectivitytunnel.cf.jp10.hana.ondemand.com	13.114.117.83, 3.114.248.68, 3.113.252.15
Japan (Tokyo) - Azure (**cf.jp20.hana.ondemand.com**)	connectivitynotification.cf.jp20.hana.ondemand.com	20.43.89.91
	connectivitycertsigning.cf.jp20.hana.ondemand.com	20.43.89.91
	connectivitytunnel.cf.jp20.hana.ondemand.com	20.43.89.91
Australia (Sydney) - AWS (**cf.ap10.hana.ondemand.com**)	connectivitynotification.cf.ap10.hana.ondemand.com	13.236.220.84, 13.211.73.244, 3.105.95.184
	connectivitycertsigning.cf.ap10.hana.ondemand.com	13.236.220.84, 13.211.73.244, 3.105.95.184

Region (Region Host)	Hosts	IP Address
	connectivitytunnel.cf.ap10.hana.ondemand.com	13.236.220.84, 13.211.73.244, 3.105.95.184
Asia Pacific (Singapore) - AWS (**cf.ap11.hana.ondemand.com**)	connectivitynotification.cf.ap11.hana.ondemand.com	3.0.9.102, 18.140.39.70, 18.139.147.53
	connectivitycertsigning.cf.ap11.hana.ondemand.com	3.0.9.102, 18.140.39.70, 18.139.147.53
	connectivitytunnel.cf.ap11.hana.ondemand.com	3.0.9.102, 18.140.39.70, 18.139.147.53
Asia Pacific (Seoul) - AWS (**cf.ap12.hana.ondemand.com**)	connectivitynotification.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12
	connectivitycertsigning.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12
	connectivitytunnel.cf.ap12.hana.ondemand.com	3.35.255.45, 3.35.106.215, 3.35.215.12
Australia (Sydney) - Azure (**cf.ap20.hana.ondemand.com**)	connectivitynotification.cf.ap20.hana.ondemand.com	20.53.99.41
	connectivitycertsigning.cf.ap20.hana.ondemand.com	20.53.99.41
	connectivitytunnel.cf.ap20.hana.ondemand.com	20.53.99.41
Singapore - Azure (**cf.ap21.hana.ondemand.com**) *Enterprise & Trial*	connectivitynotification.cf.ap21.hana.ondemand.com	20.184.61.122
	connectivitycertsigning.cf.ap21.hana.ondemand.com	20.184.61.122
	connectivitytunnel.cf.ap21.hana.ondemand.com	20.184.61.122
Canada (Montreal) - AWS (**cf.ca10.hana.ondemand.com**)	connectivitynotification.cf.ca10.hana.ondemand.com	35.182.75.101, 35.183.74.34
	connectivitycertsigning.cf.ca10.hana.ondemand.com	35.182.75.101, 35.183.74.34
	connectivitytunnel.cf.ca10.hana.ondemand.com	35.182.75.101, 35.183.74.34
China (Shanghai) - Alibaba Cloud (**cf.cn40.platform.sapcloud.cn**)	connectivitynotification.cf.cn40.platform.sapcloud.cn	139.224.7.71
	connectivitycertsigning.cf.cn40.platform.sapcloud.cn	139.224.7.71
	connectivitytunnel.cf.cn40.platform.sapcloud.cn	139.224.7.71

Region (Region Host)	Hosts	IP Address

Back to Network [page 242]

Back to Content [page 239]

Region (Region Host)	Hosts	IP Address
ABAP Environment		

> **i Note**
>
> For scenarios using the ABAP environment, include the IPs of the corresponding region below in your firewall rules, **in addition** to the IPs listed for the same region above (Cloud Foundry environment), if you use IP-based firewall rules.

Europe (Frankfurt) - AWS	*.abap.eu10.hana.ondemand.com	`18.185.129.45`, `52.29.97.247`
US East (VA) - AWS	*.abap.us10.hana.ondemand.com	`52.4.22.116`, `52.1.251.254`
Japan (Tokyo) - AWS	*.abap.jp10.hana.ondemand.com	`18.177.215.21`, `3.115.146.41`

Back to Network [page 242]

Back to Content [page 239]

Region (Region Host)	Hosts	IP Address
Neo Environment		
Europe (Rot) (**`hana.ondemand.com`**)	connectivitynotification.hana.ondemand.com	`155.56.210.83`
	connectivitycertsigning.hana.ondemand.com	`155.56.210.43`
(**`eu1.hana.ondemand.com`**)	connectivitytunnel.hana.ondemand.com	`155.56.210.84`
Europe (Frankfurt) (**`eu2.hana.ondemand.com`**)	connectivitynotification.eu2.hana.ondemand.com	`157.133.206.143`
	connectivitycertsigning.eu2.hana.ondemand.com	`157.133.205.174`
	connectivitytunnel.eu2.hana.ondemand.com	`157.133.205.233`
Europe (Amsterdam) (**`eu3.hana.ondemand.com`**)	connectivitynotification.eu3.hana.ondemand.com	`157.133.141.140`
	connectivitycertsigning.eu3.hana.ondemand.com	`157.133.141.132`
	connectivitytunnel.eu3.hana.ondemand.com	`157.133.141.141`
United States East (Ashburn) (**`us1.hana.ondemand.com`**)	connectivitynotification.us1.hana.ondemand.com	Old: `65.221.12.40` New: 157.133.18.120
	connectivitycertsigning.us1.hana.ondemand.com	Old: `65.221.12.241` New: `157.133.18.101`

Region (Region Host)	Hosts	IP Address
> **Note** Due to a network update, IP addresses for this region change as of May 24, 2020. Please make sure to include also the new IP addresses in your firewall rules if you use IP-based firewall rules.	connectivitytunnel.us1.hana.ondemand.com	Old: `65.221.12.41` New: `157.133.18.121`
United States West (Chandler) (**us2.hana.ondemand.com**)	connectivitynotification.us2.hana.ondemand.com	Old: `64.95.110.215` New: `157.133.26.25`
> **Note** Due to a network update, IP addresses for this region change as of June 14, 2020. Please make sure to include also the new IP addresses in your firewall rules if you use IP-based firewall rules.	connectivitycertsigning.us2.hana.ondemand.com	Old: `64.95.110.211` New: `157.133.26.20`
	connectivitytunnel.us2.hana.ondemand.com	Old: `64.95.110.214` New: `157.133.26.24`
United States East (Sterling) (**us3.hana.ondemand.com**)	connectivitynotification.us3.hana.ondemand.com	`169.145.118.140`
	connectivitycertsigning.us3.hana.ondemand.com	`169.145.118.132`
	connectivitytunnel.us3.hana.ondemand.com	`169.145.118.141`
US States West (Colorado Springs) (**us4.hana.ondemand.com**)	connectivitynotification.us4.hana.ondemand.com	`157.133.45.140`
	connectivitycertsigning.us4.hana.ondemand.com	`157.133.45.132`
	connectivitytunnel.us4.hana.ondemand.com	`157.133.45.141`
Australia (Sydney) (**ap1.hana.ondemand.com**)	connectivitynotification.ap1.hana.ondemand.com	`157.133.97.47`
	connectivitycertsigning.ap1.hana.ondemand.com	`157.133.97.27`
	connectivitytunnel.ap1.hana.ondemand.com	`157.133.97.46`
China (Shanghai) (**cn1.platform.sapcloud.cn**)	connectivitynotification.cn1.platform.sapcloud.cn	`157.133.194.81`
	connectivitycertsigning.cn1.platform.sapcloud.cn	`157.133.194.77`
	connectivitytunnel.cn1.platform.sapcloud.cn	`157.133.194.82`
Japan (Tokyo)	connectivitynotification.jp1.hana.ondemand.com	`157.133.150.140`

Region (Region Host)	Hosts	IP Address
(**jp1.hana.ondemand.com**)	connectivitycertsigning.jp1.hana.ondemand.com	157.133.150.132
	connectivitytunnel.jp1.hana.ondemand.com	157.133.150.141
Canada (Toronto) (**ca1.hana.ondemand.com**)	connectivitynotification.ca1.hana.ondemand.com	157.133.54.140
	connectivitycertsigning.ca1.hana.ondemand.com	157.133.54.132
	connectivitytunnel.ca1.hana.ondemand.com	157.133.54.141
Russia (Moscow) (**ru1.hana.ondemand.com**)	connectivitynotification.ru1.hana.ondemand.com	157.133.2.140
	connectivitycertsigning.ru1.hana.ondemand.com	157.133.2.132
	connectivitytunnel.ru1.hana.ondemand.com	157.133.2.141
Brazil (São Paulo) (**br1.hana.ondemand.com**)	connectivitynotification.br1.hana.ondemand.com	157.133.246.140
	connectivitycertsigning.br1.hana.ondemand.com	157.133.246.132
	connectivitytunnel.br1.hana.ondemand.com	157.133.246.141
UAE (Dubai) (**ae1.hana.ondemand.com**)	connectivitynotification.ae1.hana.ondemand.com	157.133.85.140
	connectivitycertsigning.ae1.hana.ondemand.com	157.133.85.132
	connectivitytunnel.ae1.hana.ondemand.com	157.133.85.141
KSA (Riyadh) (**sa1.hana.ondemand.com**)	connectivitynotification.sa1.hana.ondemand.com	157.133.93.140
	connectivitycertsigning.sa1.hana.ondemand.com	157.133.93.132
	connectivitytunnel.sa1.hana.ondemand.com	157.133.93.141

Back to Network [page 242]

Back to Content [page 239]

Region (Region Host)	Hosts	IP Address
Trial (Cloud Foundry Environment)		

> **Note**
>
> In the Cloud Foundry environment, IPs are controlled by the respective IaaS provider (AWS, Azure, or GCP). IPs may change due to network updates on the provider side. Any planned changes will be announced several weeks before they take effect. See also Regions.

Europe (Frankfurt) - AWS (**cf.eu10.hana.ondemand.com**)	connectivitynotification.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223
	connectivitycertsigning.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223
	connectivitytunnel.cf.eu10.hana.ondemand.com	3.124.222.77, 3.122.209.241, 3.124.208.223

Region (Region Host)	Hosts	IP Address
United States East (VA) - AWS (**cf.us10.hana.ondemand.com**)	connectivitynotification.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211
	connectivitycertsigning.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211
	connectivitytunnel.cf.us10.hana.ondemand.com	52.23.189.23, 52.4.101.240, 52.23.1.211
Singapore - Azure (**cf.ap21.hana.ondemand.com**)	connectivitynotification.cf.ap21.hana.ondemand.com	20.184.61.122
	connectivitycertsigning.cf.ap21.hana.ondemand.com	20.184.61.122
	connectivitytunnel.cf.ap21.hana.ondemand.com	20.184.61.122
Trial (Neo Environment)		
Trial (Europe only) (**hanatrial.ondemand.com**)	connectivitynotification.hanatrial.ondemand.com	155.56.219.26
	connectivitycertsigning.hanatrial.ondemand.com	155.56.219.22
	connectivitytunnel.hanatrial.ondemand.com	155.56.219.27

Back to Network [page 242]

Back to Content [page 239]

> **i Note**
>
> If you install the Cloud Connector in a network segment that is isolated from the backend systems, make sure the exposed hosts and ports are still reachable and open them in the firewall that protects them:
>
> - for HTTP, the ports you chose for the HTTP/S server.
> - for LDAP, the port of the LDAP server.
> - for RFC, it depends on whether you use an SAProuter or not and whether load balancing is used:
>   - if you use an SAProuter, it is typically configured as visible in the network of the Cloud Connector and the corresponding *routtab* is exposing all the systems that should be used.
>   - without SAProuter, you must open the application server hosts and the corresponding gateway ports (33##, 48##). When using load balancing for the connection, you must also open the message server host and port.
>
> See also Network Zones [page 250].
>
> For more information about the used ABAP server ports, see: Ports of SAP NetWeaver Application Server ABAP.

Back to Network [page 242]

Back to Content [page 239]

Related Information

Installation on Microsoft Windows OS [page 259]
Installation on Linux OS [page 262]
Installation on Mac OS X [page 265]
Recommendations for Secure Setup [page 266]
Sizing Recommendations [page 252]

1.4.5.1.1.1 Network Zones

Choose a network zone for your Cloud Connector installation.

A customer network is usually divided into multiple network zones or subnetworks according to the security level of the contained components. For example, the DMZ that contains and exposes the external-facing services of an organization to an untrusted network, usually the Internet, and there are one or more other network zones which contain the components and services provided in the company's intranet.

You can generally choose the network zone in which to set up the Cloud Connector:

- Internet access to the SAP BTP region host, either directly or via HTTPS proxy.
- Direct access to the internal systems it provides access to, which means that there is transparent connectivity between the Cloud Connector and the internal system.

The Cloud Connector can be set up either in the DMZ and operated centrally by the IT department, or set up in the intranet and operated by the appropriate line of business.

> **i Note**
>
> The internal network must allow access to the required ports; the specific configuration depends on the firewall software used.
>
> The default ports are `80` for HTTP and `443` for HTTPS. For RFC communication, you need to open a gateway port (default: `33+<instance number>` and an arbitrary message server port. For a connection to a HANA Database (on SAP BTP) via JDBC, you need to open an arbitrary *outbound* port in your network. Mail (SMTP) communication is not supported.

1.4.5.1.1.2 System Requirements

Additional system requirements for installing and running the Cloud Connector.

Supported Browsers

The browsers you can use for the Cloud Connector Administration UI are the same as those currently supported by SAPUI5. See: Browser and Platform Support.

Minimum Disk Space for Download and Installation

The minimum free disk space required to download and install a new Cloud Connector server is as follows:

- Size of downloaded Cloud Connector installation file (ZIP, TAR, MSI files): 50 MB
- Newly installed Cloud Connector server: 70 MB
- Total: 120 MB as a minimum

Additional Disk Space for Log and Configuration Files

The Cloud Connector writes configuration files, audit log files and trace files at runtime. We recommend that you reserve between 1 and 20 GB of disk space for those files.

Trace and log files are written to `<scc_dir>/log/` within the Cloud Connector `root` directory. The `ljs_trace.log` file contains traces in general, communication payload traces are stored in `traffic_trace_*.trc`. These files may be used by SAP Support to analyze potential issues. The default trace level is `Information`, where the amount of written data is generally only a few KB per day. You can turn off these traces to save disk space. However, we recommend that you don't turn off this trace completely, but that you leave it at the default settings, to allow root cause analysis if an issue occurs. If you set the trace level to `All`, the amount of data can easily reach the range of several GB per day. Use trace level `All` only to analyze a specific issue. Payload trace, however, should normally be turned off, and used only for analysis by SAP Support.

> **i Note**
>
> Regularly back up or delete written trace files to clean up the used disk space.

Audit log files are written to `/log/audit/<subaccount-name>/audit-log_<subaccount-name>_<date>.csv` within the Cloud Connector root directory. By default, only security-related events are written in the audit log. You can change the audit log level using the administration UI, as described in Manage Audit Logs [page 508].

To be compliant with the regulatory requirements of your organization and the regional laws, the audit log files must be persisted for a certain period of time for traceability purposes. Therefore, we recommend that you

back up the audit log files regularly from the Cloud Connector file system and keep the backup for the length of time required.

Related Information

Prerequisites [page 239]

1.4.5.1.2 Sizing Recommendations

When installing a Cloud Connector, the first thing you need to decide is the sizing of the installation.

This section gives some basic guidance what to consider for this decision. The provided information includes the shadow instance, which should always be added in productive setups. See also Install a Failover Instance for High Availability [page 470].

> **i Note**
>
> The following recommendations are based on current experiences. However, they are only a rule of thumb since the actual performance strongly depends on the specific environment. The overall performance of a Cloud Connector is impacted by many factors (number of hosted subaccounts, bandwidth, latency to the attached regions, network routers in the corporate network, used JVM, and others).

Restrictions

The sizing data refer to a single Cloud Connector installation.

> **i Note**
>
> Up until now, you cannot perform horizontal scaling directly. However, you can distribute the load statically by operating multiple Cloud Connector installations with different location IDs for all involved subaccounts. In this scenario, you can use multiple destinations with virtually the same configuration, except for the location ID. See also Managing Subaccounts [page 289], step 4. Alternatively, each of the Cloud Connector instances can host its own list of subaccounts without any overlap in the respective lists. Thus, you can handle more load, if a single installation risks to be overloaded.

Related Information

Hardware Setup [page 253]
Configuration Setup [page 256]

1.4.5.1.2.1 Hardware Setup

How to choose the right sizing for your Cloud Connector installation.

Regarding the hardware, we recommend that you use different setups for master and shadow. One dedicated machine should be used for the master, another one for the shadow. Usually, a shadow instance takes over the master role only temporarily. During most of its lifetime, in the shadow state, it needs less resources compared to the master.

If the master instance is available again after a downtime, we recommend that you switch back to the actual master.

> **i** Note
>
> The sizing recommendations refer to the overall load across all subaccounts that are connected via the Cloud Connector. This means that you need to accumulate the expected load of all subaccounts and should not only calculate separately per subaccount (taking the one with the highest load as basis).

Related Information

Sizing for the Master Instance [page 253]
Sizing for the Shadow Instance [page 255]

1.4.5.1.2.1.1 Sizing for the Master Instance

Learn more about the basic criteria for the sizing of your Cloud Connector master instance.

For the master setup, keep in mind the expected load for communication between the SAP BTP and on-premise systems. The setups listed below differ in a mostly qualitative manner, without hard limits for each of them.

> **i** Note
>
> The mentioned sizes are considered as minimal configuration, larger ones are always ok. In general, the more applications, application instances, and subaccounts are connected, the more competition will exist for the limited resources on the machine.

Installation Size (S, M, L)	CPU (x86_64)	Machine Memory / Heap / Direct Memory	Disk Space
S: The expected load is small (up to **1 million requests** per day, request concurrency and size is in average low) and only a **few subaccounts** with **some applications** are connected. In addition, only a **few service channels** are used, only **small data amount is replicated** to cloud systems. Setup can be a done in a virtual or physical machine.	2 cores 2.6 GHz	4GB RAM / 1GB / 2GB	10GB
M : The expected load is medium (up to **10 million requests** per day, request concurrency and size is in average medium) and **multiple subaccounts** with **multiple applications** are connected. In addition, **many service channels** are used, and a **medium data amount is replicated** to cloud systems. We recommend that you do the setup in a virtual or physical machine. A virtual machine should be on a host without overcommitted resources.	4 cores 3.0 Ghz	16GB RAM / 4GB / 8GB	20GB

Installation Size (S, M, L)	CPU (x86_64)	Machine Memory / Heap / Direct Memory	Disk Space
L: The expected load is large (**more than 10 million requests** per day, request concurrency and size is in average medium or high) and multiple subaccounts with multiple applications are connected. In addition, many service channels are used, and a **large data amount is replicated** to cloud systems concurrently. We recommend that you do the setup in a virtual or physical machine. A virtual machine should be on a host without overcommitted resources.	8 cores 3.0 Ghz	32GB RAM / 8GB / 16GB	40GB

Particularly the **heap size** is critical. If you size it too low for the load passing the Cloud Connector, at some point the Java Virtual Machine will execute full GCs (garbage collections) more frequently, blocking the processing of the Cloud Connector completely for multiple seconds, which massively slows down overall performance. If you experience such situations regularly, you should increase the heap size in the Cloud Connector UI (choose *Configuration* > *Advanced* > *JVM*). See also Configure the Java VM [page 459].

> **i Note**
>
> You should use the same value for both `<initial heap size>` and `<maximum heap size>`.

1.4.5.1.2.1.2 Sizing for the Shadow Instance

Learn more about the basic criteria for the sizing of your Cloud Connector shadow instance (high availability mode).

The shadow installation is typically not used in standard situations and hence does not need the same sizing, assuming that the time span in which it takes over the master role is limited.

> **i Note**
>
> The shadow only acts as master, for example, during an upgrade or when an abnormal situation occurs on the master machine, and either the Cloud Connector or the full machine on OS level needs to be restarted.

While being in the shadow state, the resource consumption is very low, especially in productive environments, where typically only few configuration changes are required. Therefore, the machine sizing can usually be smaller than the one for the master. However, if you want to mitigate the risk of a longer outage of the master machine, you should increase the sizing of the shadow up to the master size:

Master	Shadow
S	**S** installation as virtual machine
M	**S** installation (with double memory) as virtual or physical machine **M** installation as virtual or physical machine for risk mitigation
L	**M** installation as virtual or physical machine **L** installation as virtual or physical machine for risk mitigation

1.4.5.1.2.2 Configuration Setup

Choose the right connection configuration options to improve the performance of the Cloud Connector.

This section provides detailed information how you can adjust the configuration to improve overall performance. This is typically relevant for an M or L installation (see Hardware Setup [page 253]). For S installations, the default configuration will probably be sufficient to handle the traffic.

To change the connection parameters, proceed as follows:

- As of Cloud Connector 2.11, you can configure the number of physical connections through the Cloud Connector UI. See also Configure Tunnel Connections [page 458].
- In versions prior to 2.11, you have to modify the configuration files with an editor and restart the Cloud Connector to activate the changes.

In general, the Cloud Connector tunnel is multiplexing multiple virtual connections over a single physical connection. Thus, a single connection can handle a considerable amount of traffic. However, increasing the maximum number of physical connections allows you to make use of the full available bandwidth and to minimize latency effects.

If the bandwidth limit of your network is reached, adding additional connections doesn't increase the througput, but will only consume more resources.

> **i Note**
>
> Different network access parameters may impact and limit your configuration options: if the access to an external network is a 1 MB line with an added latency of 50 ms, you will not be able to achieve the same data volumes like with a 10 GB line with an added latency of < 1 ms. However, even if the line is good, for example 10 GB, but with an added latency of 100 ms, the performance might still be bad.

Optimal configuration strongly depends on your actual scenarios. A good approach is to try out different settings, if the current performance does not meet your expectations.

Related Information

On-Demand To On-Premise Connections [page 257]
On-Premise To On-Demand Connections (Service Channels) [page 258]

1.4.5.1.2.2.1 On-Demand To On-Premise Connections

Configure the physical connections for on-demand to on-premise calls in the Cloud Connector.

Adjusting the number of physical connections for this direction is possible both globally in the Cloud Connector UI (*Configuration* > *Advanced*), and for individual communication partners on cloud side (*On-Demand To On-Premise* > *Applications*). For an application/instance in the SAP BTP Neo environment, you can define settings even per Java application or HANA instance.

Connections are established for each defined and connected subaccount. The current number of opened connections is visible in the Cloud Connector UI via *<Subaccount>* > *Cloud Connections*. For Neo applications with multiple processes, the configured connections are established per process, which lets you use lower overall values for such a connection.

The global default is 1 physical connection per connected subaccount. This value is used across all subaccounts hosted by the Cloud Connector instance and applies for all communication partners, if no specific value is set (*On-Demand To On-Premise* > *Applications*).

To set application-specific values, choose *Subaccount* > *Cloud To On-Premise*, tab *Applications*, section *Tunnel Connection Limits*. Here you can define the number of physical connections per application.

In general, the default should be sufficient for applications with low traffic. If you expect medium traffic for most applications, it may be useful to set the default value to 2, instead of specifying individual values per application.

The following simple rule helps you to decide, whether an individual setting for a concrete application is required:

- Per 20 threads in one process executing requests to on-premise systems, provide 1 physical connection.
- If the request or response net size is larger than 250k, make sure to add an additional connection per 2 of such clients.

> **i Note**
>
> An exact traffic forecast is difficult to achieve. It requires a deep understanding of the use case and of the possible future load generated by different applications. For this reason, we recommend that you focus on subsequent configuration adjustments, using the Cloud Connector monitoring tools to recognize bottlenecks in time, and adjust Cloud Connector configuration accordingly.

Example

For an application in the SAP BTP Neo environment, requests to on-premise systems are executed in each application thread. The expected usage is 100 concurrent users. In average, about 3 of those users are triggering a remote call to an on-premise system that returns about 400k. That is, for the number of threads, you should use 5 physical connections, for the 3 clients sending larger amounts add an additional 2, which sums up to 7 connections.

Tunnel Worker Threads

In addition to the number of connections, you can configure the number of `<Tunnel Worker Threads>`. This value should be at least equal to the maximum of all individual application tunnel connections in all subaccounts, to have at least 1 thread available for each connection that can process incoming requests and outgoing responses.

Protocol Processor Worker Threads

The value for `<Protocol Processor Worker Threads>` is mainly relevant if RFC is used as protocol. Since its communication model towards the ABAP system is a blocking one, each thread can handle only one call at a time and cannot be shared. Hence, you should provide 1 thread per 5 concurrent RFC requests.

> **i** Note
>
> The longer the RFC execution time in the backend, the more threads you should provide. Threads can be reused only after the response of a call was returned to SAP BTP.

1.4.5.1.2.2.2 On-Premise To On-Demand Connections (Service Channels)

Configure the number of physical connections for a Cloud Connector service channel.

Service channels let you configure the number of physical connections to the communication partner on cloud side, see Using Service Channels [page 443]. The default is 1. This value is used as well in versions prior to Cloud Connector 2.11, which did not offer a configuration option for each service channel. You should define the number of connections depending on the expected number of clients and, with lower priority, depending on the size of the exchanged messages.

If there is only a single RFC client for an S/4HANA Cloud channel or only a single HANA client for a HANA DB on SAP BTP side, increasing the number doesn't help, as each virtual connection is assigned to one physical connection. The following simple rule lets you to define the required number of connections per service channel:

- Per 10 concurrent clients, use one physical connection.

- If the transferred net data size is larger than 500k per request, make sure to add an additional connection per 2 of such clients.

Example

For a HANA system in the SAP BTP, data is replicated using 18 concurrent clients in the on-premise network. In average, about 5 of those clients are regularly sending 600k. For the number of clients, you should use 2 physical connections, for the 5 clients sending larger amounts add an additional 3, which sums up to 5 connections.

1.4.5.1.3 Installation on Microsoft Windows OS

Installing the Cloud Connector on a Microsoft Windows operating system.

Context

You can choose between a simple `portable` variant of the Cloud Connector and the MSI-based `installer`. The `installer` is the generally recommended version that you can use for both developer and productive scenarios. It lets you, for example, register the Cloud Connector as a Windows service and this way automatically start it after machine reboot.

> → Tip
>
> If you are a developer, you might want to use the `portable` variant as you can run the Cloud Connector after a simple unzip (archive extraction). You might want to use it also if you cannot perform a full installation due to lack of permissions, or if you want to use multiple versions of the Cloud Connector simultaneously on the same machine.

Prerequisites

- You have either of the following 64-bit operating systems: Windows 7, Windows 8.1, Windows 10, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016, or Windows Server 2019.
- You have downloaded either the `portable` variant as ZIP archive for Windows, or the MSI `installer` from the SAP Development Tools for Eclipse page.
- You must install Microsoft Visual Studio C++ 2013 runtime libraries (vcredist_x64.exe). For more information, see Visual C++ Redistributable Packages for Visual Studio 2013 .

> **Note**
>
> Even if you have a more recent version of the Microsoft Visual C++ runtime libraries, you still must install the Microsoft Visual Studio C++ 2013 libraries.

- Java 8 must be installed. In case you want to use SAP JVM, you can download it from the SAP Development Tools for Eclipse page.
- When using the `portable` variant, the environment variable `<JAVA_HOME>` must be set to the Java installation directory, so that the `bin` subdirectory can be found. Alternatively, you can add the relevant `bin` subdirectory to the `<PATH>` variable.

Portable Scenario

1. Extract the `<sapcc-<version>-windows-x64.zip>` ZIP file to an arbitrary directory on your local file system.
2. Set the environment variable JAVA_HOME to the installation directory of the JDK that you want to use to run the Cloud Connector. Alternatively, you can add the `bin` subdirectory of the JDK installation directory to the PATH environment variable.
3. Go to the Cloud Connector installation directory and start it using the `go.bat` batch file.
4. Continue with the **`Next Steps`** section.

> **Note**
>
> The Cloud Connector is not started as a service when using the portable variant, and hence will not automatically start after a reboot of your system. Also, the portable version does not support the automatic upgrade procedure.

Installer Scenario

1. Start the `<sapcc-<version>-windows-x64.msi>` installer by double-clicking it.
2. The installer informs you that you are now guided through the installation process. Choose *Next>*.
3. Navigate to the desired installation directory for your Cloud Connector and choose *Next>*. When doing the installation in the context of an upgrade, make sure you choose the previous installation directory again.
4. You can choose the port on which the administration UI is reachable. Either leave the default **`8443`** or choose a different port if needed. Then, choose *Next>*.
5. Select the JDK to be used for running the Cloud Connector. The installer displays a list of all usable JDKs that are installed on your machine. If the needed JDK is not listed in the drop-down box (for example, if it's an SAP JVM that is not registered in the Windows registry upon installation), you can browse to its installation directory and select it. We recommend that you use an up-to-date Java **8** installation to run the Cloud Connector.
6. Decide whether the Cloud Connector should be started immediately after finishing the setup. Then, choose *Next>*.

7. To start the installation, press the *Next>* button again.
8. After successful installation, choose *Close*.
9. Continue with the **`Next Steps`** section.

> **i** Note
>
> The Cloud Connector is started as a Windows service in the productive use case. Therefore, installation requires administration permissions. After installation, manage this service under *Control Panel* > *Administrative Tools* > *Services*. The service name is `Cloud Connector` (formerly named `Cloud Connector 2.0`). Make sure the service is executed with a user that has limited privileges. Typically, privileges allowed for service users are defined by your company policy. Adjust the folder and file permissions to be manageable by only this user and system administrators.

On Windows, the file `scc_service.log` is created and used by the Microsoft MSI installer (during Cloud Connector installation), and by the `scchost.exe` executable, which registers and runs the Windows service if you install the Cloud Connector as a Windows background job.

This log file is only needed if a problem occurs during Cloud Connector installation, or during creation and start of the Windows service, in which the Cloud Connector is running. You can find the file in the `log` folder of your Cloud Connector installation directory.

Starting the Cloud Connector

After installation, the Cloud Connector is registered as a Windows service that is configured to be started automatically after a system reboot. You can start and stop the service via shortcuts on the desktop ("Start Cloud Connector" and "Stop Cloud Connector"), or by using the `Windows Services` manager and look for the service **`SAP Cloud Connector`**.

Access the Cloud Connector administration UI at *https://localhost:<port>*, where the default port is **`8443`** (but this port might have been modified during the installation).

Next Steps

1. Open a browser and enter: **`https://<hostname>:8443`**. `<hostname>` is the host name of the machine on which you have installed the Cloud Connector. If you access the Cloud Connector locally from the same machine, you can simply enter **`localhost`**.
2. Continue with the initial configuration of the Cloud Connector, see Initial Configuration [page 278].

Related Information

(Optional) Install SAP JVM [page 846]

Recommendations for Secure Setup [page 266]

[Deprecated] Replace the Default SSL Certificate [page 272]

1.4.5.1.4 Installation on Linux OS

Installing the Cloud Connector on a Linux operating system.

Context

You can choose between a simple `portable` variant of the Cloud Connector and the RPM-based `installer`. The `installer` is the generally recommended version that you can use for both the developer and the productive scenario. It registers, for example, the Cloud Connector as a daemon service and this way automatically starts it after machine reboot.

> → Tip
>
> If you are a developer, you might want to use the `portable` variant as you can run the Cloud Connector after a simple "`tar -xzof`" execution. You also might want to use it if you cannot perform a full installation due to missing permissions for the operating system, or if you want to use multiple versions of the Cloud Connector simultaneously on the same machine.

Prerequisites

- You have either of the following 64-bit operating systems: SUSE Linux Enterprise Server 11, 12, or 15, or Redhat Enterprise Linux 6, 7, or 8.
- The supported platforms are `x64` and `ppc64le`, represented below by the variable `<platform>`. Variable `<arch>` is `x86_64` or `ppc64le` respectively.
- You have downloaded either the `portable` variant as `tar.gz` archive for Linux or the RPM `installer` contained in the ZIP for Linux, from SAP Development Tools for Eclipse.
- Java 8 must be installed. If you want to use SAP JVM, you can download an up-to-date version from SAP Development Tools for Eclipse as well. Use the following command to install it:

  ```
  rpm -i sapjvm-<version>-linux-<platform>.rpm
  ```

  If you want to check the JVM version installed on your system, use the following command:

  ```
  rpm -qa | grep jvm
  ```

  When installing it using the RPM package, the Cloud Connector will detect it and use it for its runtime.
- When using the **`tar.gz`** archive, the environment variable `<JAVA_HOME>` must be set to the Java installation directory, so that the `bin` subdirectory can be found. Alternatively, you can add the Java installation's `bin` subdirectory to the `<PATH>` variable.

Portable Scenario

1. Extract the *tar.gz* file to an arbitrary directory on your local file system using the following command:

   ```
   tar -xzof sapcc-<version>-linux-<platform>.tar.gz
   ```

   > **i Note**
   >
   > If you use the parameter "o", the extracted files are assigned to the user ID and the group ID of the user who has unpacked the archive. This is the default behavior for users other than the `root` user.

2. Go to this directory and start the Cloud Connector using the `go.sh` script.
3. Continue with the **`Next Steps`** section.

> **i Note**
>
> In this case, the Cloud Connector is not started as a daemon, and therefore will not automatically start after a reboot of your system. Also, the `portable` version does not support the automatic upgrade procedure.

Installer Scenario

1. Extract the *sapcc-<version>-linux-<platform>.zip* archive to an arbitrary directory by using the following the command:

   ```
   unzip sapcc-<version>-linux-<platform>.zip
   ```

2. Go to this directory and install the extracted RPM using the following command. You can perform this step only as a `root` user.

   ```
   rpm -i com.sap.scc-ui-<version>.<arch>.rpm
   ```

3. Continue with the **`Next Steps`** section.

In the productive case, the Cloud Connector is started as a daemon. If you need to manage the daemon process, execute:

```
System V init distributions: service scc_daemon stop|restart|start|status
systemd distributions: systemctl stop|restart|start|status scc_daemon
```

> **⚠ Caution**
>
> When adjusting the Cloud Connector installation (for example, restoring a backup), make sure the RPM package management is synchronized with such changes. If you simply replace files that do not fit to the information stored in the package management, lifecycle operations (such as upgrade or uninstallation) might fail with errors. Also, the Cloud Connector might get into unrecoverable state.
>
> **Example**: After a file system restore, the system files represent Cloud Connector 2.3.0 but the RPM package management "believes" that version 2.4.3 is installed. In this case, commands like **`rpm -U`** and **`rpm -e`** do not work as expected. Furthermore, avoid using the `--force` parameter as it may lead to an unpredictable state with two versions being installed concurrently, which is not supported.

Extending the Daemon (as of Cloud Connector version 2.12.3)

When using SNC for encrypting RFC communication, it might be required to provide some settings, for example, environment variables that must be visible for the Cloud Connector process. To achieve this, you must store a file named `scc_daemon_extension.sh` in the installation directory of the Cloud Connector (`/opt/sap/scc`), containing all commands needed for initialization without a shebang.

Example (SAP CommonCryptoLibrary requires SECUDIR to be set):

Sample Code

```
export SECUDIR=/path/to/psefile
```

To activate it, you must reinstall the daemon. Make sure JAVA_HOME is set to the JVM used. Then execute the following command to reinstall the daemon:

```
System V init distributions: /opt/sap/scc/daemon.sh reinstall
systemd distributions: /opt/sap/scc/daemon.sh reinstallSystemd
```

The daemon extension will survive Cloud Connector version updates.

Starting the Cloud Connector

After installation via RPM manager, the Cloud Connector process is started automatically and registered as a daemon process, which ensures the automatic restart of the Cloud Connector after a system reboot.

To start, stop, or restart the process explicitly, open a command shell and use the following commands, which require `root` permissions:

```
System V init distributions: service scc_daemon start|stop|restart
systemd distributions: systemctl start|stop|restart scc_daemon
```

Next Steps

1. Open a browser and enter: **`https://<hostname>:8443`**. `<hostname>` is the host name of the machine on which you installed the Cloud Connector.
   If you access the Cloud Connector locally from the same machine, you can simply enter **`localhost`**.
2. Continue with the initial configuration of the Cloud Connector, see Initial Configuration [page 278].

Related Information

Recommendations for Secure Setup [page 266]
[Deprecated] Replace the Default SSL Certificate [page 272]

1.4.5.1.5 Installation on Mac OS X

Installing the Cloud Connector on a Mac OS X operating system.

Prerequisites

> **i Note**
>
> Mac OS X is not supported for productive scenarios. The developer version described below must not be used as productive version.

- You have either of the following 64-bit operating systems: Mac OS X 10.7 (Lion), Mac OS X 10.8 (Mountain Lion), Mac OS X 10.9 (Mavericks), Mac OS X 10.10 (Yosemite), or Mac OS X 10.11 (El Capitan), Mac OS X 10.12 (Sierra), Mac OS X 10.13 (High Sierra), or Mac OS X 10.14 (Mojave).
- You have downloaded the `tar.gz` archive for the developer use case on Mac OS X from SAP Development Tools for Eclipse.
- Java 8 must be installed. If you want to use SAP JVM, you can download it from SAP Development Tools for Eclipse as well.
- Environment variable `<JAVA_HOME>` must be set to the Java installation directory so that the `bin` subdirectory can be found. Alternatively, you can add the Java installation's `bin` subdirectory to the `<PATH>` variable.

Procedure

1. Extract the `tar.gz` file to an arbitrary directory on your local file system using the following command:

   ```
   tar -xzof sapcc-<version>-macosx-x64.tar.gz
   ```

2. Go to this directory and start Cloud Connector using the `go.sh` script.
3. Continue with the **`Next Steps`** section.

   > **i Note**
   >
   > The Cloud connector is not started as a daemon, and therefore will not automatically start after a reboot of your system. Also, the Mac OS X version of Cloud Connector does not support the automatic upgrade procedure.

Next Steps

1. Open a browser and enter: **`https://<hostname>:8443`**. `<hostname>` is the host name of the machine on which you installed the Cloud Connector.
   If you access the Cloud Connector locally from the same machine, you can simply enter **`localhost`**.

2. Continue with the initial configuration of the Cloud Connector, see Initial Configuration [page 278].

Related Information

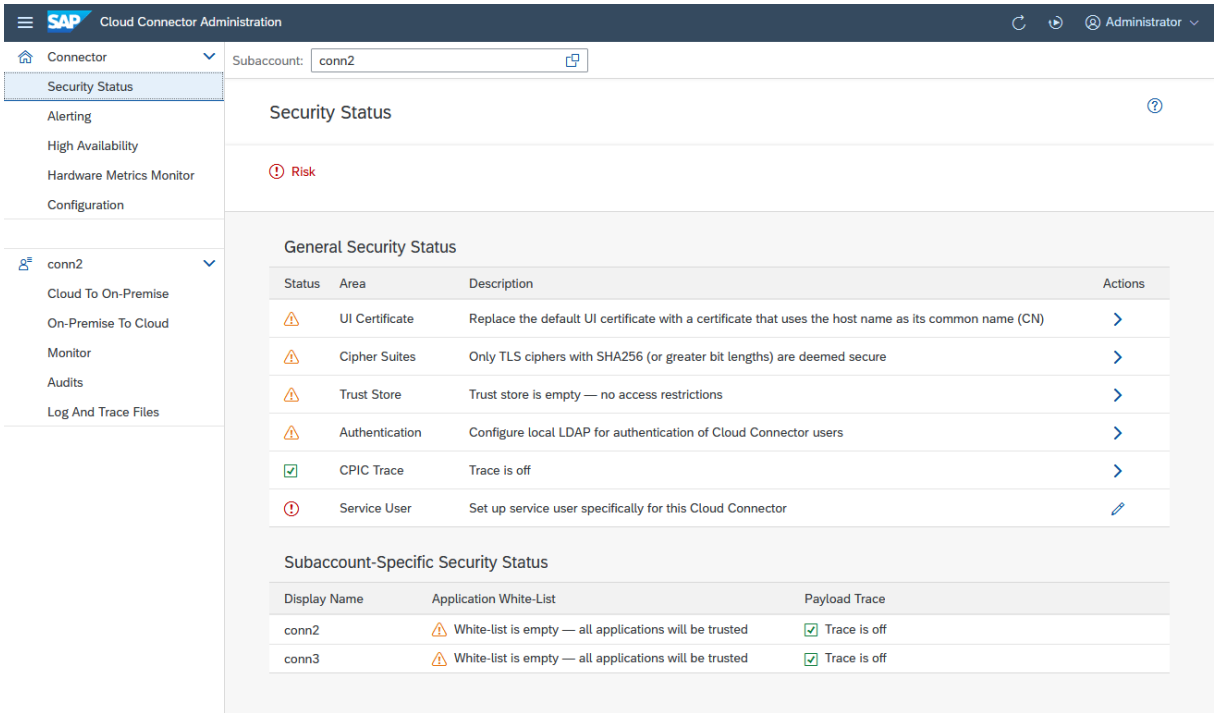
Recommendations for Secure Setup [page 266]
[Deprecated] Replace the Default SSL Certificate [page 272]

1.4.5.1.6 Recommendations for Secure Setup

For the Connectivity service and the Cloud Connector, you should apply the following guidelines to guarantee the highest level of security for these components.

Security Status

From the *Connector* menu, choose *Security Status* to access an overview showing potential security risks and the recommended actions.



The *General Security Status* addresses security topics that are subaccount-independent.

- Choose any of the *Actions* icons in the corresponding line to navigate to the UI area that deals with that particular topic and view or edit details.

  > **i Note**
  >
  > Navigation is not possible for the last item in the list (*Service User*).

- The service user is specific to the Windows operating system (see Installation on Microsoft Windows OS [page 259] for details) and is only visible when running the Cloud Connector on Windows. It cannot be accessed or edited through the UI. If the service user was set up properly, choose *Edit* and check the corresponding checkbox.

The *Subaccount-Specific Security Status* lists security-related information for each and every subaccount.

> **i Note**
>
> The security status only serves as a reminder to address security issues and shows if your installation complies with all recommended security settings.

Password Policy

Upon installation, the Cloud Connector provides an initial user name and password for the administration UI, and forces the user (`Administrator`) to change the password. You must change the password immediately after installation.

The connector itself does not check the strength of the password. You should select a strong password that cannot be guessed easily.

> **i Note**
>
> To enforce your company's password policy, we recommend that you configure the Administration UI to use an LDAP server for authorizing access to the UI.

UI Access

The Cloud Connector administration UI can be accessed remotely via HTTPS. The connector uses a standard X.509 self-signed certificate as SSL server certificate. You can exchange this certificate with a specific certificate that is trusted by your company. See [Deprecated] Replace the Default SSL Certificate [page 272].

> **i Note**
>
> Since browsers usually do not resolve *localhost* to the host name whereas the certificate usually is created under the host name, you might get a certificate warning. In this case, simply skip the warning message.

OS-Level Access

The Cloud Connector is a security-critical component that handles the external access to systems of an isolated network, comparable to a reverse proxy. We therefore recommend that you restrict the access to the operating system on which the Cloud Connector is installed to the minimal set of users who would administrate the Cloud Connector. This minimizes the risk of unauthorized users getting access to credentials, such as certificates stored in the secure storage of the Cloud Connector.

We also recommend that you use the machine to operate only the Cloud Connector and no other systems.

Administrator Privileges

To log on to the Cloud Connector administration UI, the `Administrator` user of the connector must not have an operating system (OS) user for the machine on which the connector is running. This allows the OS administrator to be distinguished from the Cloud Connector administrator. To make an initial connection between the connector and a particular SAP BTP subaccount, you need an SAP BTP user with the required permissions for the related subaccount. We recommend that you separate these roles/duties (that means, you have separate users for Cloud Connector administrator and SAP BTP).

> **i** Note
>
> We recommend that only a small number of users are granted access to the machine as *root* users.

Hard Drive Encryption

Hard drive encryption for machines with a Cloud Connector installation ensures that the Cloud Connector configuration data cannot be read by unauthorized users, even if they obtain access to the hard drive.

Supported Protocols

Currently, the protocols HTTP and RFC are supported for connections between the SAP BTP and on-premise systems when the Cloud Connector and the Connectivity service are used. The whole route from the application virtual machine in the cloud to the Cloud Connector is always SSL-encrypted.

The route from the connector to the back-end system can be SSL-encrypted or SNC-encrypted. See Configure Access Control (HTTP) [page 331] and Configure Access Control (RFC) [page 338].

Audit Log on OS Level

We recommend that you turn on the audit log on operating system level to monitor the file operations.

Audit Log on Cloud Connector Level

The Cloud Connector audit log must remain switched on during the time it is used with productive systems. The default audit level is **`SECURITY`**. Set it to **`ALL`** if required by your company policy. The administrators who are responsible for a running Cloud Connector must ensure that the audit log files are properly archived, to conform to the local regulations. You should switch on audit logging also in the connected back-end systems.

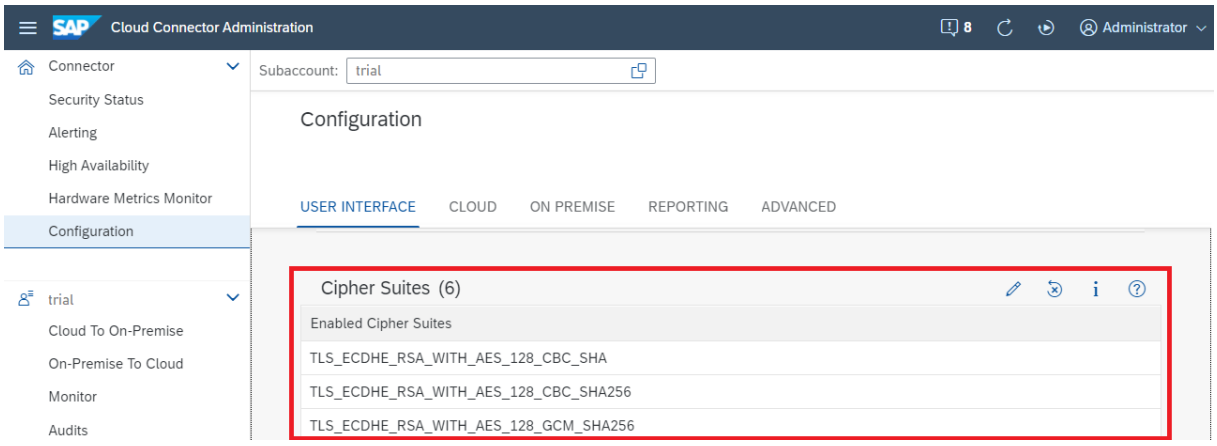
Encryption Ciphers

By default, all available encryption ciphers are supported for HTTPS connections to the administration UI. However, some of them may not conform to your security standards and therefore should be excluded:

1. From the main menu, choose *Configuration* and select the tab *User Interface*, section *Cipher Suites*. By default, all available ciphers are marked as selected.
2. Choose the *Remove* icon to unselect the ciphers that do not meet your security requirements.

> **i Note**
>
> We recommend that you revert the selection to the default (all ciphers selected) whenever you plan to switch to another JVM. As the set of supported ciphers may differ, the selected ciphers may not be supported by the new JVM. In that case the Cloud Connector does not start anymore. You need to fix the issue manually by adapting the file `default-server.xml` (cp. attribute ciphers, see section *Accessing the Cloud Connector Administrator UI* above). After switching the JVM, you can adjust the list of eligible ciphers.

3. Choose *Save*.



Related Information

Connectivity via Reverse Proxy [page 543]
Security [page 519]

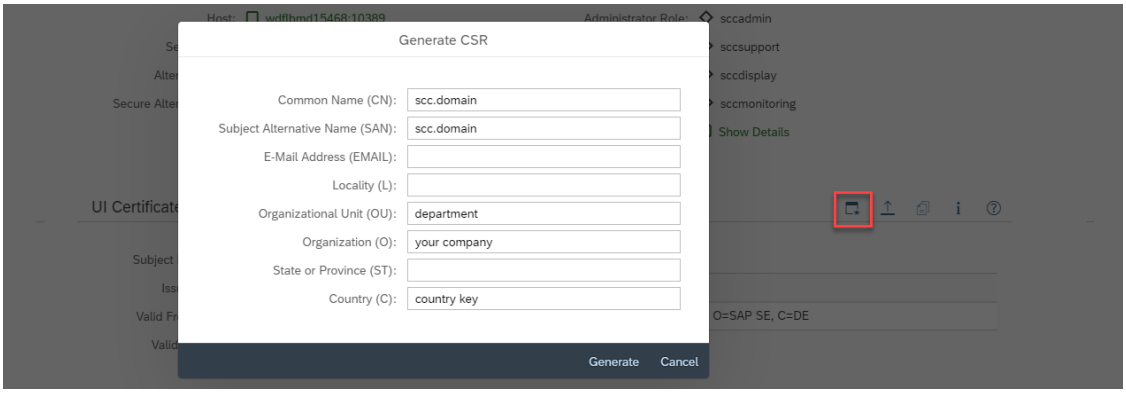
1.4.5.1.7 Recommended: Exchange UI Certificates in the Administration UI

By default, the Cloud Connector includes a self-signed UI certificate. It is used to encrypt the communication between the browser-based user interface and the Cloud Connector itself. For security reasons, however, you should replace this certificate with your own one to let the browser accept the certificate without security warnings.

Procedure

Master Instance

1. From the main menu, choose *Configuration* and go to the *User Interface* tab.
2. In the *UI Certificate* section, start a certificate signing request procedure by choosing the icon *Generate a Certificate Signing Request*.
3. In the pop-up *Generate CSR*, specify a subject fitting to your host name.
   For host matching, you should use the available names within the `subjectAlternativeName` (SAN) extension, see RFC 2818 . A check verifies whether the host matches one of the entries in the SAN extension.
   Choose either of the procedures below, according to your Cloud Connector version:
   1. **Procedure up to Cloud Connector version 2.12.0**

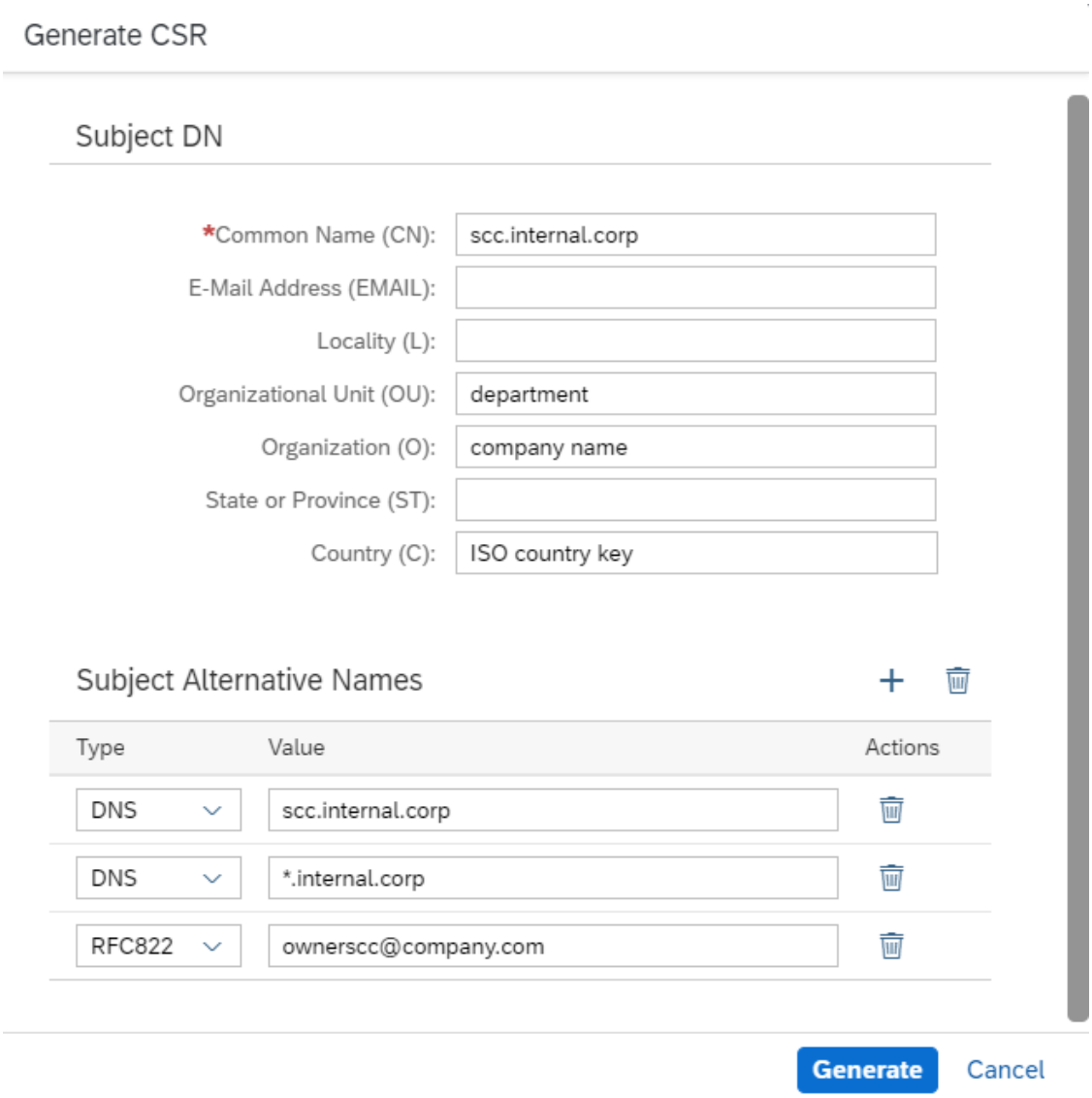


   The field `<SAN>` allows a simple value as well as formatted complex values:
   - A simple value is treated as DNS name, for example, **`xyz.sap.com`** means that the allowed host is xyz.sap.com.
   - `<SAN>` also allows a list of DNS names, IPs (4 byte or IPv6), URIs, and RFC 822 names (for example, e-mail addresses).

     > **i Note**
     >
     > In this case, the field `<SAN>` contains key:value pairs separated by ';'. ';' must not be used in a value.

     Example for a complex SAN value : `DNS:localhost;DNS:*.sap.com;IP:10.20.30.40;IP:fe80::78cc:a107:dcf2:73fe%13`.

2. **Procedure as of Cloud Connector version 2.12.1**



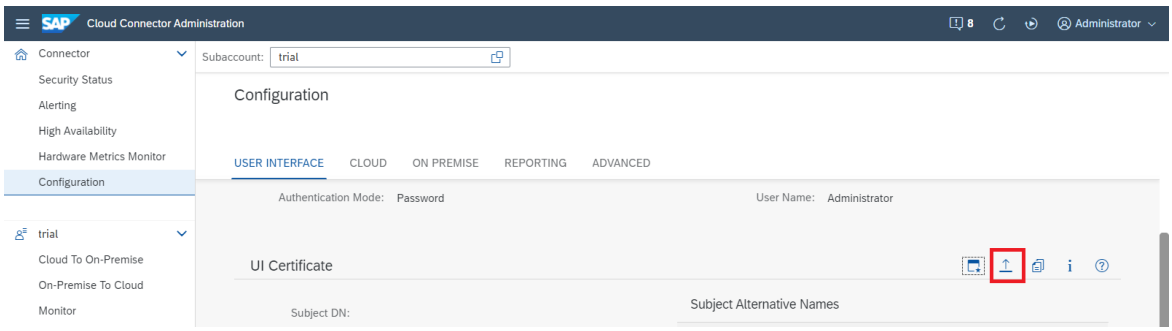
This new version simplifies the SAN mamagement. The CSR generation dialog now separates SAN values from the Subject DN of the certificate by introducing two sections. In the new section **Subject Alternative Names**, you can add additional values easily by pressing the *Add* button. Choose one or more of the following SAN types and provide the matching values:

- DNS: a specific host name (for example, `www.sap.com`) or a wildcard hostname (for example, `*.sap.com`).
- IP: an IPv4 or IPv6 address.
- RFC822 : an example for this type of value is a simple email address: for example, `donotreply@sap.com`.
- URI: a URI for which the certificate should be valid.

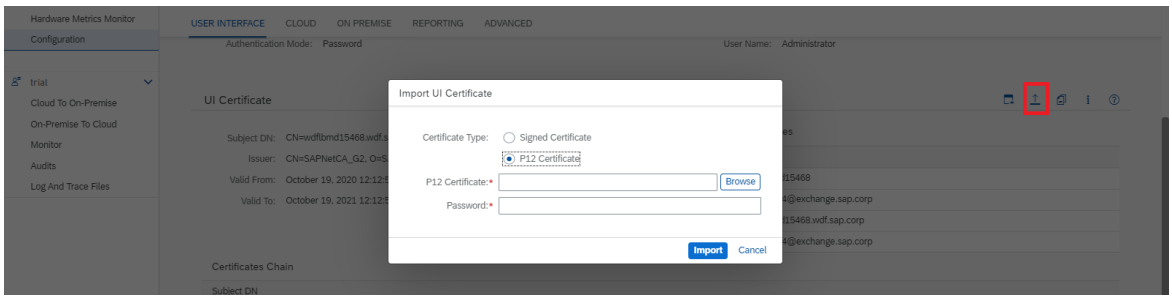
4. Press *Generate*.
5. You are prompted to save the signing request in a file. The content of the file is the signing request in PEM format.
   The signing request must be provided to a Certificate Authority (CA) - either one within your company or another one you trust. The CA signs the request and the returned response should be stored in a file.

> **Note**
>
> The response should be either an X.509 certificate or a PKCS#7 in PEM format.

6. To import the signing response, choose the *Upload* icon.



As of Cloud Connector version 2.13, you can also upload an existing PKCS#12 certificate directly (instead of generating a CSR).



7. Select *Browse* to locate the file and then choose the *Import* button.
8. Review the certificate details that are displayed.
9. Restart the Cloud Connector to activate the new certificate.

Shadow Instance

In a High Availability setup, perform the same operation on the shadow instance.

1.4.5.1.7.1 [Deprecated] Replace the Default SSL Certificate

> **Note**
>
> This procedure only applies for Cloud Connector versions prior to 2.13. You can now use the UI-based procedure instead, see Recommended: Exchange UI Certificates in the Administration UI [page 270].

Overview

By default, the Cloud Connector includes a self-signed UI certificate. It is used to encrypt the communication between the browser-based user interface and the Cloud Connector itself. For security reasons, however, you should replace this certificate with your own certificate so that the browser accepts the certificate without security warnings.

Up to version **2.5.2**, for this purpose, you need to know the password of the Cloud Connector's Java keystore. This password is generated during installation and then kept in an encrypted secure storage area. To obtain the password, follow the steps described below.

> **i Note**
>
> As of version **2.6.0**, you can easily replace the default certificate within the Cloud Connector administration UI . See Recommended: Exchange UI Certificates in the Administration UI [page 270].

> **⚠ Caution**
>
> The Cloud Connector's keystore may contain a certificate used in the High Availability setup. This certificate has the alias "**ha**". Any changes on it or removal would cause a disruption of communication between the shadow and the master instance, and therefore to a failed procedure. We recommend that you replace the keystore on both the master and shadow server before establishing the connection between the two instances.

Procedure

You can read the password by executing the following command:

- on Microsoft Windows OS:

```
java -cp <scc_install_dir>\plugins\com.sap.scc.rt*.jar -
Djava.library.path=<scc_install_dir>\auditor com.sap.scc.jni.SecStoreAccess -
path <scc_install_dir>\scc_config -p
```

- on Linux OS:

```
java -cp /opt/sap/scc/plugins/com.sap.scc.rt*.jar -
Djava.library.path=/opt/sap/scc/auditor com.sap.scc.jni.SecStoreAccess -
path /opt/sap/scc/scc_config -p
```

> **i Note**
>
> Memorize the keystore password, as you will need it for later operations. See related links.

Make sure you go to directory `/opt/sap/scc/config` before executing the commands described in the following procedures.

> **i Note**
>
> For a detailed description of the *keytool* tool, see http://docs.oracle.com/javase/7/docs/technotes/tools/solaris/keytool.html .

Related Information

Recommended: Exchange UI Certificates in the Administration UI [page 270]
[Deprecated] Use a Self-Signed Certificate [page 274]
[Deprecated] Use Certificates Signed by a Trusted Certificate Authority [page 275]

1.4.5.1.7.2 [Deprecated] Use a Self-Signed Certificate

Generate a self-signed certificate for special purposes like, for example, a demo setup.

Context

> **i Note**
>
> As of Cloud Connector 2.10 you can generate self-signed certificates also from the administration UI. See Configure a CA Certificate for Principal Propagation [page 304] and Initial Configuration (HTTP) [page 285]. In this case, the steps below are not required.

If you want to use a simple, self-signed certificate, follow the procedure below.

> **i Note**
>
> The parameter values in the following section are examples.
>
> The server configuration delivered by SAP uses the same password for key store (option \-storepass) and key (option \-keypass) under alias `tomcat`.

Procedure

1. Remove the current default certificate:

   ```
   keytool -delete -alias tomcat -keystore ks.store -storepass <password>
   ```

2. Generate a certificate:

   ```
   keytool -genkey -v -keyalg RSA -alias tomcat -keypass <password> -keystore
   ks.store -storepass <password> -dname "CN=SCC, OU=<YourCompany>,
   O=<YourCompany>"
   ```

3. Self-sign it - you will be prompted for the keypass password defined in step 2:

   ```
   keytool -selfcert -v -alias tomcat -storepass <password> -keystore ks.store
   ```

1.4.5.1.7.3 [Deprecated] Use Certificates Signed by a Trusted Certificate Authority

Use a signed certificate by a trusted certificate authority (CA) to increase the security level when running the Cloud Connector.

> **i Note**
>
> This procedure only applies for Cloud Connector versions prior to 2.13. You can now use the UI-based procedure instead, see Recommended: Exchange UI Certificates in the Administration UI [page 270].

Before starting the procedure, note the following:

- The parameter values of the following steps are only examples.
- We recommend that you use a signed certificate by a trusted CA, because it is more secure than a self-signed certificate.
- For your convenience, you can set the generated password as environment variable, like in the command below, and then use **$PASS** as a password:
  ```
  export PASS=`<the release-dependent command as given in the parent page>`
  ```
- The keytool supports `delete` and `changealias` commands. If the Cloud Connector SSL certificate is changed on a running instance, we recommend that you prepare a new certificate under a temporary alias. Once everything is ready, you change the alias.

Procedure

> **i Note**
>
> If you already have a signed certificate produced by a trusted certificate authority (CA), skip steps 1,2, and 4, and only follow the instructions provided in step 3.

1. Generate your key pair if you start fresh:

   ```
   keytool -genkey -v -keyalg RSA -alias tomcat -keypass <password> -keystore
   ks.store -storepass <password> -dname "CN=SCC, OU=<YourCompany>,
   O=<YourCompany>"
   ```

   Alternatively, you may reuse an existing key store.
2. Create a local certificate signing request (CSR):

   ```
   keytool -certreq -keyalg RSA -alias tomcat -keypass <password> -keystore
   ks.store -storepass <password> -file <csr-file-name>
   ```

   You now have a file called `<csr-file-name>` that you can submit to the certificate authority. In return, you get a certificate.
3. Import the certificate chain that you obtained from your trusted CA:

   ```
   keytool -import -alias root -keystore ks.store -storepass <password> -
   trustcacerts -file <filename_of_the_certificate_chain>
   ```

> **Note**
>
> If you already have a signed certificate produced by a trusted certificate authority (CA), continue with the following steps (skipping 1,2, and 4):
>
> - Create a backup copy of the keystore file `ks.store`.
> - Import your keystore with option `-importkeystore` (documented here) into the Cloud Connector keystore `ks.store`.
>
> ```
> keytool -importkeystore -srckeystore mypfxfile.pfx -srcstoretype pkcs12
> -destkeystore ks.store -deststoretype JKS -srcstorepass <your_pw> -
> deststorepass <pw_from_sec_store> - srcalias <your_alias> -destalias
> tomcat -srckeypass <your_pw> -destkeypass <pw_from_sec_store>
> ```

4. Import your new certificate:

```
keytool -import -alias tomcat -keystore ks.store -storepass <password> -file
<your_certificate_filename>
```

The password is created at installation time and stored in the secure storage. Thus, only applications with access can read the password. You can read password using Java:

```
jar -xf /opt/sap/scc/dropins/scc/plugins/com.sap.scc.tomcat.utils*.jar  lib/
libsapsecstore4j.so
java -cp /opt/sap/scc/dropins/scc/plugins/com.sap.scc.tomcat.utils*.jar -
Djava.library.path=./lib/ com.sap.mw.scc.util.SecStoreAccess -show
```

You might need to adapt the configuration if you want to use another key storage file or change the current configuration (HTTPS port, authentication type, SSL protocol, and so on). You can find the SSL configuration in the *Connector* section of the file:

- Microsoft Windows OS: `<install_dir>\config_master\org.eclipse.gemini.web.tomcat \default-server.xml`
- Linux OS: `/opt/sap/scc/config_master/org.eclipse.gemini.web.tomcat/default-server.xml`

> **Note**
>
> We recommend that you do not modify the configuration unless you have expertise in this area.

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true"
    keystoreFile="config/ks.store" keystorePass="${jks.password}" keyPass="$
{jks.password}" keyAlias="tomcat"
    truststoreFile="config/ks.store" truststorePass="${jks.password}"
    clientAuth="want" sslProtocol="TLS"
    compression="on" compressionMinSize="1024"
noCompressionUserAgents="gozilla,traviata,*MSIE 6.*"
    compressableMimeType="text/html,text/xml,text/plain,text/javascript,text/
css,text/json,application/x-javascript,application/javascript,application/json"/>
```

Related Information

For more information about configuring SSL, see http://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html#SSL_and_Tomcat .

1.4.5.2 Configuration

Configure the Cloud Connector to make it operational for connections between your SAP BTP applications and on-premise systems.

Topic	Description
Initial Configuration [page 278]	After installing the Cloud Connector and starting the Cloud Connector daemon, you can log on and perform the required configuration to make your Cloud Connector operational.
Managing Subaccounts [page 289]	How to connect SAP BTP subaccounts to your Cloud Connector.
Authenticating Users against On-Premise Systems [page 300]	Basic authentication and principal propagation (user propagation) are the authentication types currently supported by the Cloud Connector.
Configure Access Control [page 330]	Configure access control or copy the complete access control settings from another subaccount on the same Cloud Connector.
Configuration REST APIs [page 355]	Configure a newly installed Cloud Connector (initial configuration, subaccounts, access control) using the configuration REST API.
Configure an On-Premise User Store [page 441]	Configure applications running on SAP BTP to use your corporate LDAP server as a user store.
Using Service Channels [page 443]	Service channels provide access from an external network to certain services on SAP BTP, which are not exposed to direct access from the Internet.
Configure Trust [page 452]	Set up an allowlist for trusted cloud applications and a trust store for on-premise systems in the Cloud Connector.
Connect DB Tools to SAP HANA via Service Channels [page 447]	How to connect database, BI, or replication tools running in the on-premise network to a HANA database on SAP BTP using the service channels of the Cloud Connector.
Configure Domain Mappings for Cookies [page 455]	Map virtual and internal domains to ensure correct handling of cookies in client/server communication.
Configure Solution Management Integration [page 457]	Activate Solution Management reporting in the Cloud Connector.
Configure Tunnel Connections [page 458]	Adapt connectivity settings that control the throughput by choosing the appropriate limits (maximal values).
Configure the Java VM [page 459]	Adapt the JVM settings that control memory management.

Topic	Description
Configuration Backup [page 460]	Backup and restore your Cloud Connector configuration.

1.4.5.2.1 Initial Configuration

After installing and starting the Cloud Connector, log on to the administration UI and perform the required configuration to make your Cloud Connector operational.

Tasks

Prerequisites [page 278]

Log on to the Cloud Connector [page 279]

Change your Password and Choose Installation Type [page 280]

Set up Connection Parameters and HTTPS Proxy [page 281]

Establish Connections to SAP BTP [page 284]

Prerequisites

- You have assigned one of these roles/role collections to the subaccount user that you use for initial Cloud Connector setup, depending on the SAP BTP environment in which your subaccount is running:

  > **i** Note
  >
  > For the **Cloud Foundry** environment, you must know on which cloud management tools feature set (A or B) your account is running. For more information on feature sets, see Cloud Management Tools — Feature Set Overview.

Environment	Required Roles/Role Collections	More Information
Cloud Foundry [feature set **A**]	The user must be a *member of the global account* that the subaccount belongs to. Alternatively, you can assign the user as *Security Administrator*.	Add Members to Your Global Account Managing Security Administrators in Your Subaccount [Feature Set A]

Environment	Required Roles/Role Collections	More Information
Cloud Foundry [feature set **B**]	Assign at least one of these *default role collections* (all of them including the role `Cloud Connector Administrator`): ○ `Subaccount Administrator` ○ `Cloud Connector Administrator` ○ `Connectivity and Destination Administrator` Alternatively, you can assign a *custom role collection* to the user that includes the role `Cloud Connector Administrator`.	*Default Role Collections [Feature Set B]* Role Collections and Roles in Global Accounts, Directories, and Subaccounts [Feature Set B]
Neo	Assign at least one of these *default roles*: ○ `Cloud Connector Admin` ○ `Administrator` Alternatively, you can assign a *custom role* to the user that includes the permission `manageSCCTunnels`.	Managing Member Authorizations in the Neo Environment [page 1301]

After establishing the Cloud Connector connection, this user is not needed any more, since it serves only for initial connection setup. You may revoke the corresponding role assignment then and remove the user from the *Members* list (**Neo** environment), or from the *Users* list (**Cloud Foundry** environment).

> **i** Note
>
> If the Cloud Connector is installed in an environment that is operated by SAP, SAP provides a user that you can add as member in your SAP BTP subaccount and assign the required role.

- We strongly recommend that you read and follow the steps described in Recommendations for Secure Setup [page 266]. For operating the Cloud Connector securely, see also Security Guidelines [page 526].

Back toTasks [page 278]

Log on to the Cloud Connector

To administer the Cloud Connector, you need a Web browser. To check the list of supported browsers, see Prerequisites and Restrictions → section *Browser Support*.

1. In a Web browser, enter: **`https://<hostname>:<port>`**

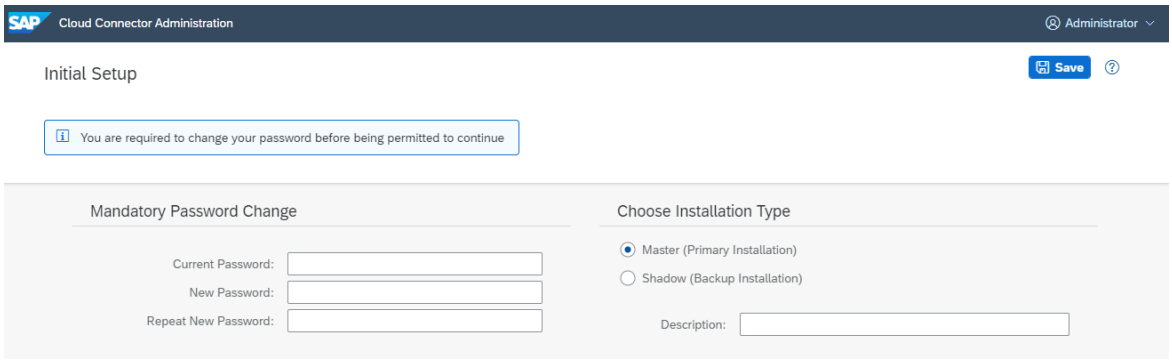
- `<hostname>` refers to the machine on which the Cloud Connector is installed. If installed on your machine, you can simply enter **`localhost`**.
- `<port>` is the Cloud Connector port specified during installation (the default port is **`8443`**).

2. On the logon screen, enter **`Administrator`** / **`manage`** (case sensitive) for `<User Name>` / `<Password>`.

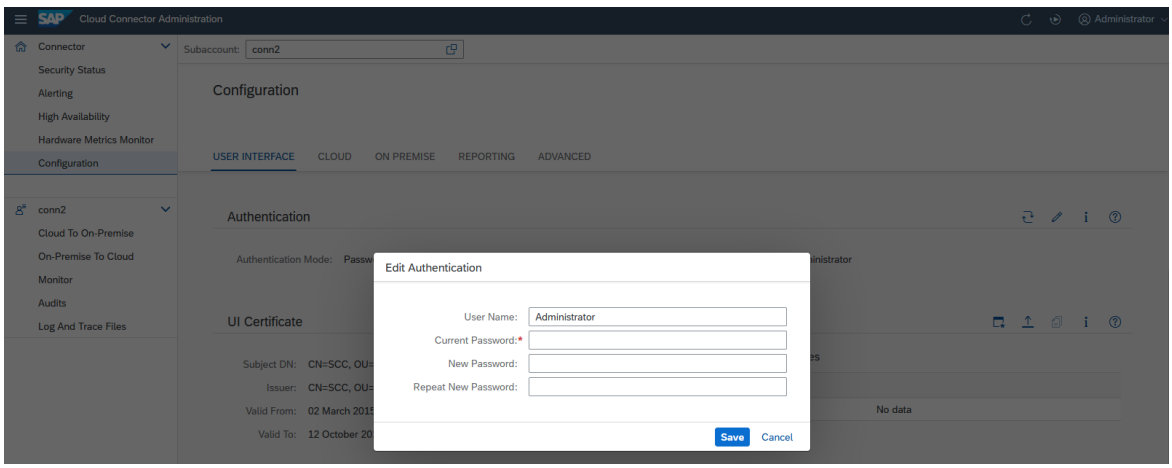
Back toTasks [page 278]

Change your Password and Choose Installation Type

1. When you first log in, you must change the password before you continue, regardless of the installation type you have chosen.
2. Choose between master and shadow installation. Use *Master* if you are installing a single Cloud Connector instance or a main instance from a pair of Cloud Connector instances. See Install a Failover Instance for High Availability [page 470].



3. You can edit the password for the **`Administrator`** user from *Configuration* in the main menu, tab *User Interface*, section *Authentication*:

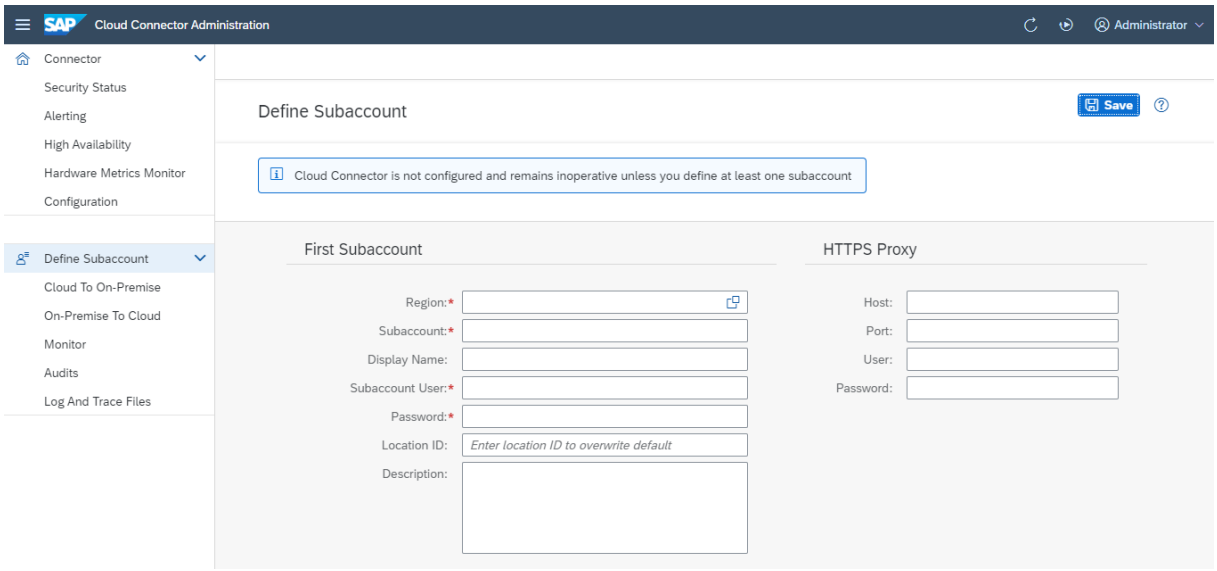


> ### i Note
>
> User name and password cannot be changed at the same time. If you want to change the user name, you must enter only the current password in a first step. Do not enter values for `<New Password>` or `<Repeat New Password>` when changing the user name. To change the password in second step, enter the old password, the new one, and the repeated (new) password, but leave the user name unchanged.

Back toTasks [page 278]

Set up Connection Parameters and HTTPS Proxy

When logging in for the first time, the following screen is displayed every time you choose an option from the main menu that requires a configured subaccount:



If your internal landscape is protected by a firewall that blocks any outgoing TCP traffic, you must specify an HTTPS proxy that the Cloud Connector can use to connect to SAP BTP. Normally, you must use the same proxy settings as those being used by your standard Web browser. The Cloud Connector needs this proxy for two operations:

- Download the correct connection configuration corresponding to your subaccount ID in SAP BTP.
- Establish the SSL tunnel connection from the Cloud Connector user to your SAP BTP subaccount.

> **i Note**
>
> If you want to skip the initial configuration, you can click the  icon in the upper right corner. You might need this in case of connectivity issues shown in your logs. You can add subaccounts later as described in Managing Subaccounts [page 289].

The Cloud Connector collects the following required information for your subaccount connection:

1. For `<Region>`, specify the SAP BTP region that should be used. You can choose it from the drop-down list, see Regions.

   > **i Note**
   >
   > You can also configure a region yourself, if it is not part of the standard list. Either insert the region host manually, or create a custom region, as described in Configure Custom Regions [page 300].

2. For `<Subaccount>`, `<Subaccount User>` and `<Password>`, enter the values you obtained when you registered your subaccount on SAP BTP.

   > **i Note**
   >
   > For the **Neo** environment, enter the subaccount's **technical name** in the field `<Subaccount>`, not the subaccount ID.

   You can also add a new subaccount user with the role `Cloud Connector Admin` in the SAP BTP cockpit and use the new user and password.

   > **i Note**
   >
   > The Cloud Connector does not yet support *SAP Universal ID*. Please use your S-user or P-user credentials for the `<subaccount user>` and `<password>` fields instead.
   >
   > For more information, see SAP note 3085908.

   For the **Neo** environment, see Add Members to Your Neo Subaccount [page 1299].
   For the **Cloud Foundry** environment, see Add Org Members Using the Cockpit.

   > **→ Tip**
   >
   > When using SAP Cloud Identity Services - Identity Authentication (IAS) as platform identity provider with two-factor authentication for your subaccount, you can simply append the required token to the regular password.

3. (Optional) You can define a `<Display Name>` that lets you easily recognize a specific subaccount in the UI compared to the technical subaccount name.
4. (Optional) You can define a `<Location ID>` identifying the location of this Cloud Connector for a specific subaccount. As of Cloud Connector release **2.9.0**, the location ID is used as routing information and therefore you can connect multiple Cloud Connectors to a single subaccount. If you don't specify any value for `<Location ID>`, the default is used, which represents the behavior of previous Cloud Connector versions. The location ID must be unique per subaccount and should be an identifier that can be used in a URI. To route requests to a Cloud Connector with a location ID, the location ID must be configured in the respective destinations.

   > **i Note**
   >
   > Location IDs provided in older versions of the Cloud Connector are discarded during upgrade to ensure compatibility for existing scenarios.

5. Enter a suitable proxy host from your network and the port that is specified for this proxy. If your network requires an authentication for the proxy, enter a corresponding proxy user and password. You must specify a proxy server that supports SSL communication (a standard HTTP proxy does not suffice).

   > **i Note**
   >
   > These settings strongly depend on your specific network setup. If you need more detailed information, please contact your local system administrator.

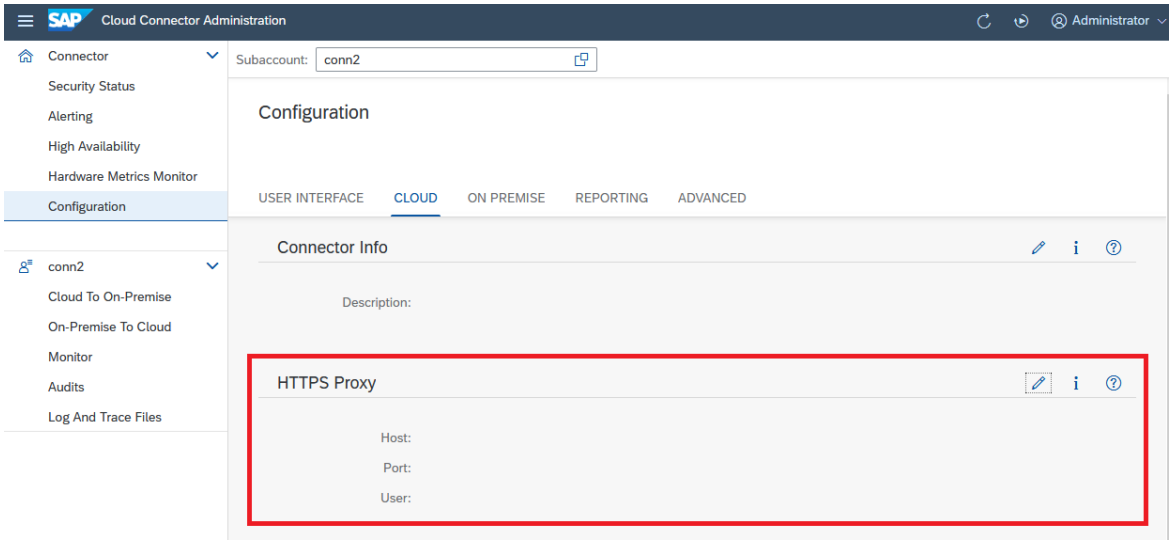
6. (Optional) You can provide a `<Description>` (free-text) of the subaccount that is shown when choosing the *Details* icon in the *Actions* column of the *Subaccount Dashboard*. It lets you identify the particular Cloud Connector you use.

7. Choose *Save*.

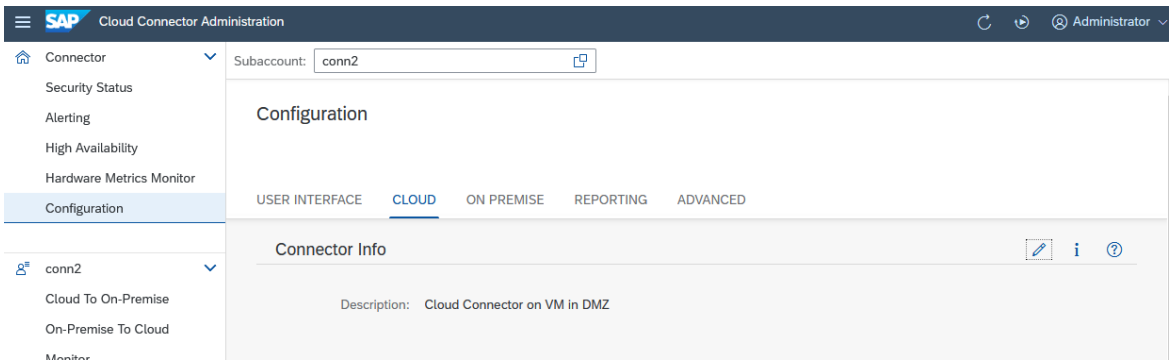
The Cloud Connector now starts a handshake with SAP BTP and attempts to establish a secure SSL tunnel to the server that hosts the subaccount in which your on-demand applications are running. However, no requests are yet allowed to pass from the cloud side to any of your internal back-end systems. To allow your on-demand applications to access specific internal back-end systems, proceed with the access configuration described in the next section.

> **i Note**
>
> The internal network must allow access to the port. Specific configuration for opening the respective port(s) depends on the firewall software used. The default ports are `80` for HTTP and `443` for HTTPS. For RFC communication, you must open a gateway port (default: `33+<instance number>` and an arbitrary message server port. For a connection to a HANA Database (on SAP BTP) via JDBC, you must open an arbitrary *outbound* port in your network. Mail (SMTP) communication is not supported.

- If you later want to change your proxy settings (for example, because the company firewall rules have changed), choose *Configuration* from the main menu and go to the *Cloud* tab, section *HTTPS Proxy*. Some proxy servers require credentials for authentication. In this case, you must provide the relevant user/password information.



- If you want to change the description for your Cloud Connector, choose *Configuration* from the main menu, go to the *Cloud* tab, section *Connector Info* and edit the description:

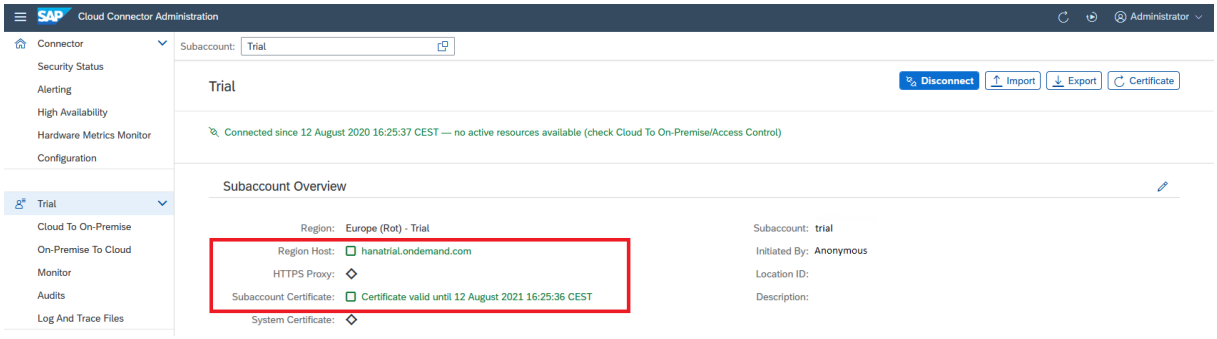


Back toTasks [page 278]

Establish Connections to SAP BTP

As soon as the initial setup is complete, the tunnel to the cloud endpoint is open, but no requests are allowed to pass until you have performed the *Access Control* setup, see Configure Access Control [page 330].

To manually close (and reopen) the connection to SAP BTP, choose your subaccount from the main menu and select the *Disconnect* button (or the *Connect* button to reconnect to SAP BTP).



- The green icon next to *Region Host* indicates that it is valid and can be reached.
- If an *HTTPS Proxy* is configured, its availability is shown the same way. In the screenshot, the grey diamond icon next to *HTTPS Proxy* indicates that connectivity is possible without proxy configuration.

In case of a timeout or a connectivity issue, these icons are yellow (warning) or red (error), and a tooltip shows the cause of the problem. *Initiated By* refers to the user that has originally established the tunnel. During normal operations, this user is no longer needed. Instead, a certificate is used to open the connection to a subaccount.

- The status of the certificate is shown next to *Subaccount Certificate*. It is shown as valid (green icon), if the expiration date is still far in the future, and turns to yellow if expiration approaches according to your alert settings. It turns red as soon as it has expired. This is the latest point in time, when you should Update the Certificate for Your Subaccount [page 296].

> **i Note**
>
> When connected, you can monitor the Cloud Connector also in the *Connectivity* section of the SAP BTP cockpit. There, you can track attributes like version, description and high availability set up. Every Cloud Connector configured for your subaccount automatically appears in the *Connectivity* section of the cockpit.

Back toTasks [page 278]

Related Information

Managing Subaccounts [page 289]
Initial Configuration (HTTP) [page 285]

Initial Configuration (RFC) [page 287]
Configuring the Cloud Connector for LDAP [page 288]
Managing Member Authorizations in the Neo Environment [page 1301]

1.4.5.2.1.1 Initial Configuration (HTTP)

Configure the Cloud Connector for HTTP communication.

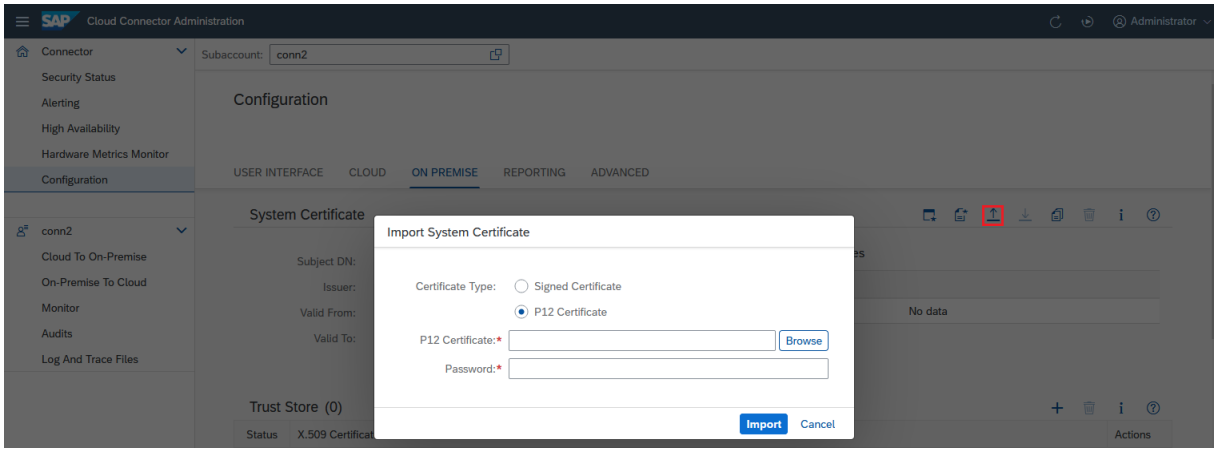
Installation of a System Certificate for Mutual Authentication

To set up a mutual authentication between the Cloud Connector and any backend system it connects to, you can import an X.509 client certificate into the Cloud Connector. The Cloud Connector then uses the so-called *system certificate* for all HTTPS requests to backends that request or require a client certificate. The CA that signed the Cloud Connector's client certificate must be trusted by all backend systems to which the Cloud Connector is supposed to connect.

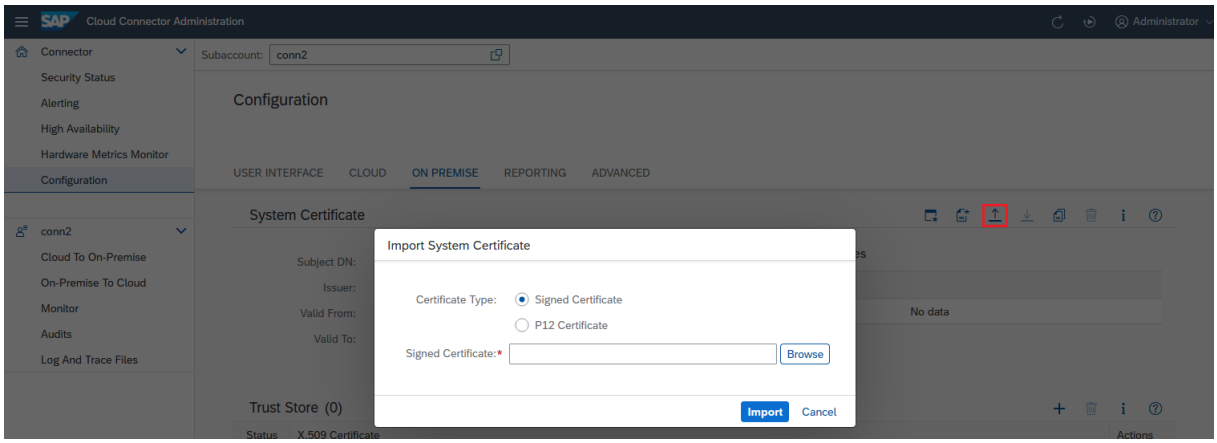
You must provide the system certificate as PKCS#12 file containing the client certificate, the corresponding private key and the CA root certificate that signed the client certificate (plus potentially the certificates of any intermediate CAs, if the certificate chain is longer than 2).

Procedure

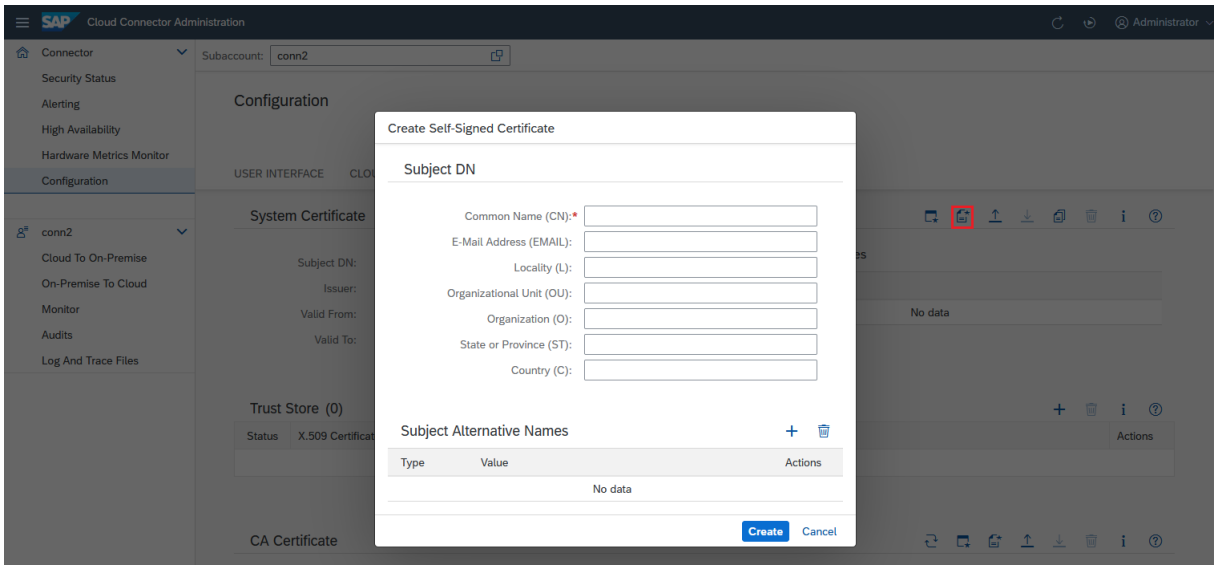
From the left panel, choose *Configuration*. On the tab *On Premise*, choose *System Certificate* > *Import a certificate* to upload a certificate and provide its password:



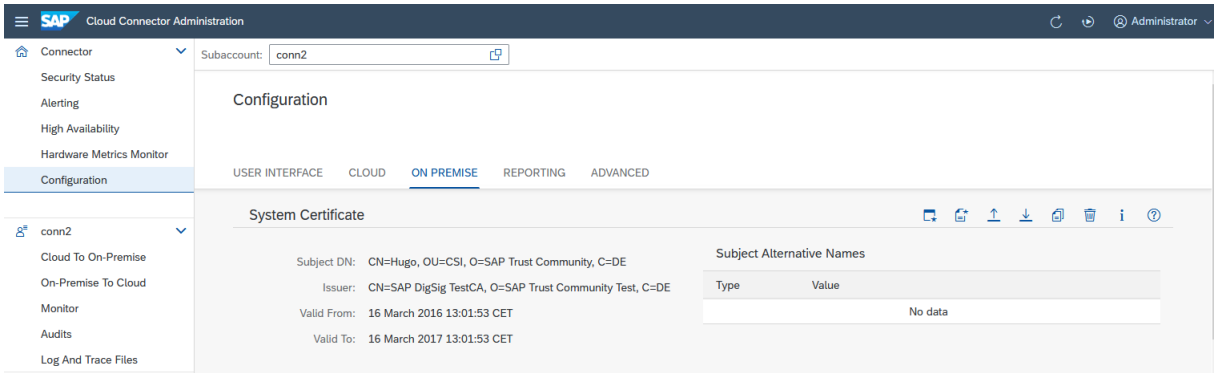
A second option is to start a *certificate signing request* procedure as described for the UI certificate in Recommended: Exchange UI Certificates in the Administration UI [page 270] and upload the resulting signed certificate.



As of version 2.10, there is a third option - generating a self-signed certificate. It might be useful if no CA is needed, for example, in a demo setup or if you want to use a dedicated CA. For this option, choose *Create and import a self-signed certificate*:



If a system certificate has been imported successfully, its distinguished name, the name of the issuer, and the validity dates are displayed:



If a system certificate is no longer required, you can delete it. To do this, use the respective button and confirm deletion. If you need the public key for establishing trust with a server, you can simply export the full chain via the *Export* button.

Related Information

Configure Access Control (HTTP) [page 331]

1.4.5.2.1.2 Initial Configuration (RFC)

Configure a Secure Network Connection (SNC) to set up the Cloud Connector for RFC communication to an ABAP backend system.

SNC Configuration for Mutual Authentication

To set up a mutual authentication between Cloud Connector and an ABAP backend system (connected via RFC), you can configure SNC for the Cloud Connector. It will then use the associated PSE for all RFC SNC requests. This means that the SNC identity, represented by this PSE, must:

- Be trusted by all backend systems to which the Cloud Connector is supposed to connect
- Play the role of a trusted external system by adding the SNC name of the Cloud Connector to the SNCSYSACL table. You can find more details in the SNC configuration documentation for the release of your ABAP system.

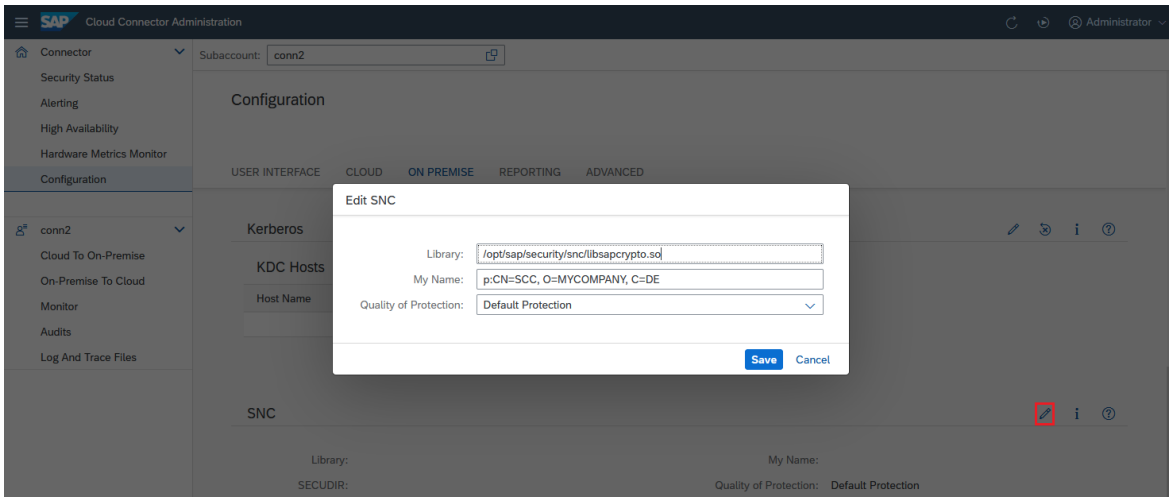
Prerequisites

You have configured your ABAP system(s) for SNC. For detailed information on configuring SNC for an ABAP system, see also Configuring SNC on AS ABAP. In order to establish trust for Principal Propagation, follow the steps described in Configure Principal Propagation for RFC [page 315].

Configuration Steps

1. Logon to the Cloud Connector
2. Choose *Configuration* from the main menu and go to tab *On Premise*, section *SNC*.
3. Enter the corresponding values in the fields *Library Name*, *My Name*, and *Quality of Protection*
4. Press *Save*.

Example:



- *Library Name*: Provides the location of the SNC library you are using for the Cloud Connector.

  > **i Note**
  >
  > Bear in mind that you must use one and the same security product on both sides of the communication.

- *My Name*: The SNC name that identifies the Cloud Connector. It represents a valid scheme for the SNC implementation that is used.
- *Quality of Protection*: Determines the level of protection that you require for the connectivity to the ABAP systems.

> **i Note**
>
> When using **CommonCryptoLibrary** as SNC implementation, note 1525059 will help you to configure the PSE to be associated with the user running the Cloud Connector process.

Related Information

Configure Principal Propagation for RFC [page 315]

1.4.5.2.1.3 Configuring the Cloud Connector for LDAP

Configure the Cloud Connector to support LDAP in different scenarios (cloud applications using LDAP or Cloud Connector authentication).

Prerequisites

You have installed the Cloud Connector and done the basic configuration:

Installation [page 238]

Initial Configuration [page 278]

Steps

When using LDAP-based user management, you have to confgure the Cloud Connector to support this feature. Depending on the scenario, you need to perform the following steps:

Scenario 1: Cloud applications using LDAP for authentication. Configure the destination of the LDAP server in the Cloud Connector: Configure Access Control (LDAP) [page 344].

Scenario 2: Internal Cloud Connector user management. Activate LDAP user management in the Cloud Connector: Use LDAP for Authentication [page 466].

1.4.5.2.2 Managing Subaccounts

Add and connect your SAP BTP subaccounts to the Cloud Connector.

> **i** Note
>
> This topic refers to subaccount management in the Cloud Connector. If you are looking for information about managing subaccounts on SAP BTP (Cloud Foundry or Neo environment), see
>
> - Account Administration (**Cloud Foundry** environment)
> - Administration and Operations, Neo Environment [page 1259]

Context

As of version **2.2**, you can connect to several subaccounts within a single Cloud Connector installation. Those subaccounts can use the Cloud Connector concurrently with different configurations. By selecting a subaccount from the drop-down box, all tab entries show the configuration, audit, and state, specific to this subaccount. In case of audit and traces, cross-subaccount info is merged with the subaccount-specific parts of the UI.

> **i** Note
>
> We recommend that you group only subaccounts with the same qualities in a single installation:

- Productive subaccounts should reside on a Cloud Connector that is used for productive subaccounts only.
- Test and development subaccounts can be merged, depending on the group of users that are supposed to deal with those subaccounts. However, the preferred logical setup is to have separate development and test installations.

Prerequisites

You have assigned one of these roles/role collections to the subaccount user that you use for initial Cloud Connector setup, depending on the SAP BTP environment in which your subaccount is running:

> **i** Note
>
> For the **Cloud Foundry** environment, you must know on which cloud management tools feature set (A or B) your account is running. For more information on feature sets, see Cloud Management Tools — Feature Set Overview.

Environment	Required Roles/Role Collections	More Information
Cloud Foundry [feature set **A**]	The user must be a *member of the global account* that the subaccount belongs to. Alternatively, you can assign the user as *Security Administrator*.	Add Members to Your Global Account Managing Security Administrators in Your Subaccount [Feature Set A]
Cloud Foundry [feature set **B**]	Assign at least one of these *default role collections* (all of them including the role `Cloud Connector Administrator`): • `Subaccount Administrator` • `Cloud Connector Administrator` • `Connectivity and Destination Administrator` Alternatively, you can assign a *custom role collection* to the user that includes the role `Cloud Connector Administrator`.	*Default Role Collections [Feature Set B]* Role Collections and Roles in Global Accounts, Directories, and Subaccounts [Feature Set B]

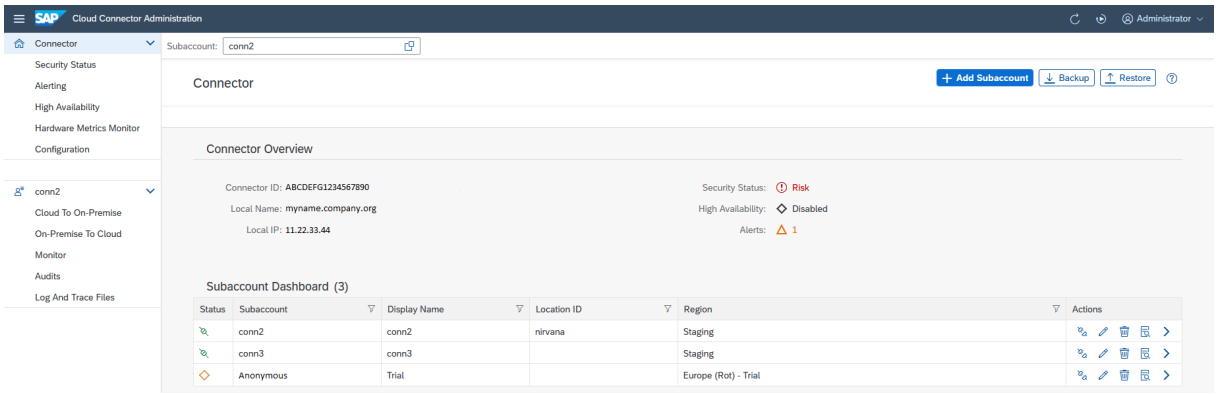
Environment	Required Roles/Role Collections	More Information
Neo	Assign at least one of these *default roles*: • `Cloud Connector Admin` • `Administrator` Alternatively, you can assign a *custom role* to the user that includes the permission `manageSCCTunnels`.	Managing Member Authorizations in the Neo Environment [page 1301]

After establishing the Cloud Connector connection, this user is not needed any more, since it serves only for initial connection setup. You may revoke the corresponding role assignment then and remove the user from the *Members* list.

> **i Note**
>
> If the Cloud Connector is installed in an environment that is operated by SAP, SAP provides a user that you can add as member in your SAP BTP subaccount and assign the required role.

Subaccount Dashboard

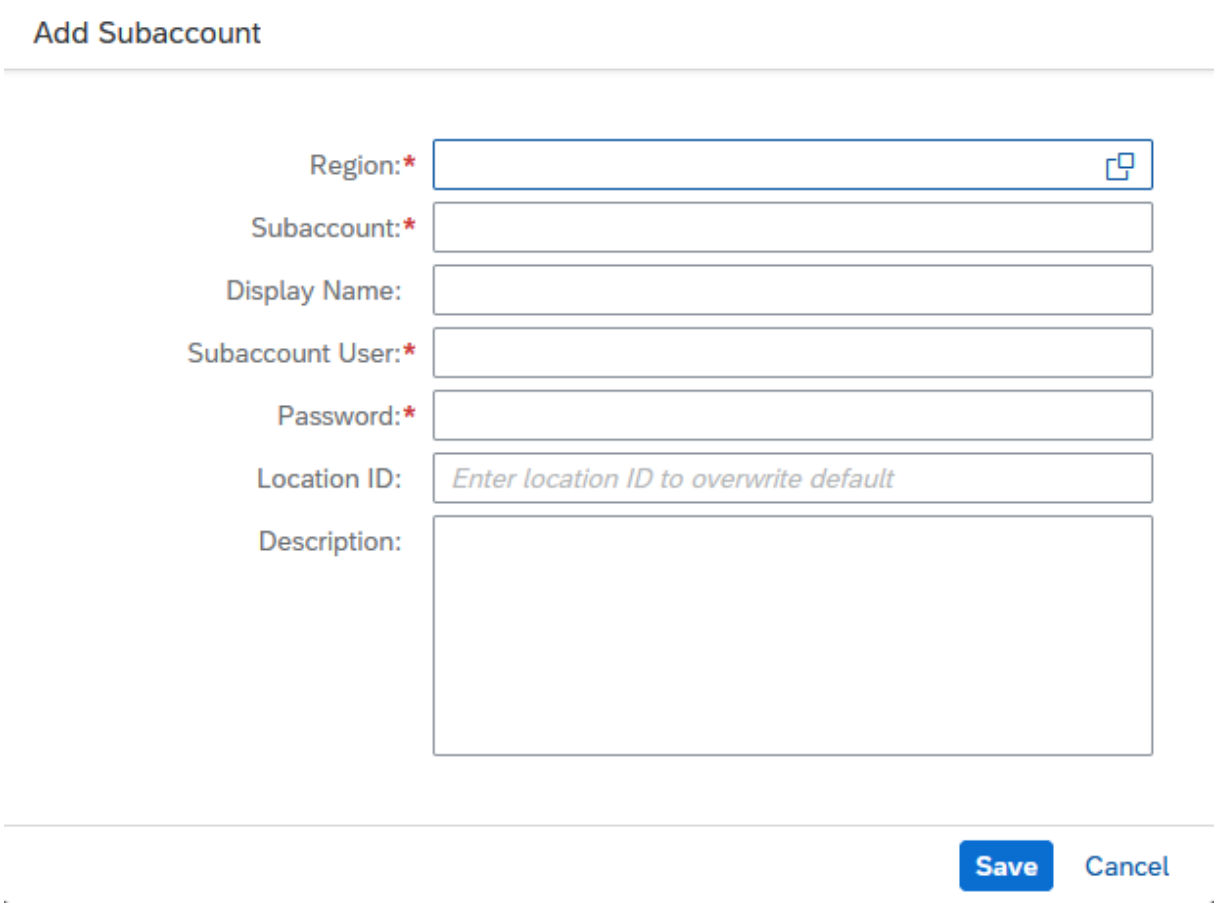
In the subaccount dashboard (choose your *Subaccount* from the main menu), you can check the state of all subaccount connections managed by this Cloud Connector at a glance.



In the screenshot above, the `test1` subaccount is already connected, but has no active resources exposed. The `test2` subaccount is currently disconnected.

The dashboard also lets you disconnect or connect the subaccounts by choosing the respective button in the *Actions* column.

If you want to connect an additional subaccount to your on-premise landscape, choose the *Add Subaccount* button. A dialog appears, which is similar to the *Initial Configuration* operation when establishing the first connection.



Procedure

1. The `<Region>` field specifies the SAP BTP region that should be used, for example, **`Europe (Rot)`**. Choose the one you need from the drop-down list.

   > → Remember
   >
   > The available regions and region domains depend on the SAP BTP environment you are using. For more information, see Regions (Cloud Foundry and ABAP environment) or Regions and Hosts Available for the Neo Environment.

   > → Tip
   >
   > You can also configure a region yourself, if it is not part of the standard list. Either insert the region host manually, or create a custom region, as described in Configure Custom Regions [page 300].

2. For `<Subaccount>` and `<Subaccount User>` (user/password), enter the values you obtained when you registered your account on SAP BTP.

   > i Note
   >
   > If your subaccount is on **Cloud Foundry**, you must enter the subaccount **ID** as `<Subaccount>`, rather than its actual (technical) name. For information on getting the subaccount ID, see Find Your

Subaccount ID (Cloud Foundry Environment) [page 299]. As `<Subaccount User>` you must provide your **`Login E-mail`** instead of a user ID.

For the **Neo** environment, enter the subaccount's **technical name** in the field `<Subaccount>`, not the subaccount ID.

Alternatively, you can add a new subaccount user in the SAP BTP cockpit, assign the required authorization (see section **Prerequisites** above), and use the new user and password.

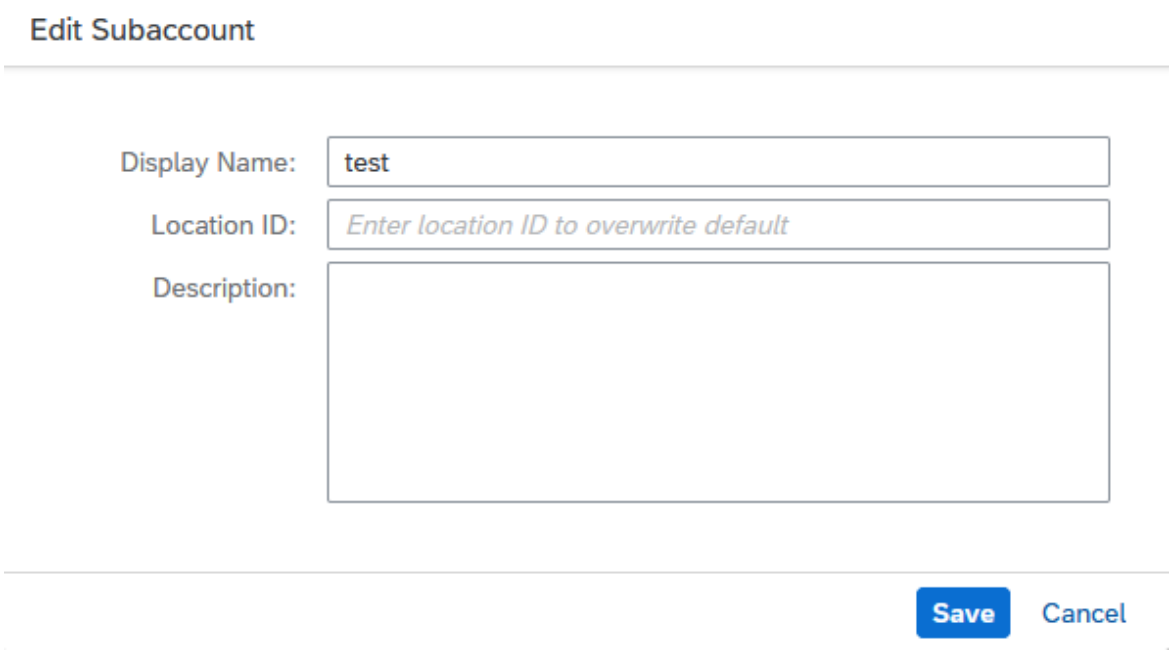
> **i** Note
>
> The Cloud Connector does not yet support *SAP Universal ID*. Please use your S-user or P-user credentials for the `<subaccount user>` and `<password>` fields instead.
>
> For more information, see SAP note 3085908.

> **→** Tip
>
> When using SAP Cloud Identity Services - Identity Authentication (IAS) as platform identity provider with two-factor authentication for your subaccount, you can simply append the required token to the regular password.

3. (Optional) You can define a `<Display Name>` that allows you to easily recognize a specific subaccount in the UI compared to the technical subaccount name.
4. (Optional) You can define a `<Location ID>` that identifies the location of this Cloud Connector for a specific subaccount. As of Cloud Connector release 2.9.0, the location ID is used as routing information and therefore you can connect multiple Cloud Connectors to a single subaccount. If you don't specify any value for `<Location ID>`, the default is used, which represents the behavior of previous Cloud Connector versions. The location ID must be unique per subaccount and should be an identifier that can be used in a URI. To route requests to a Cloud Connector with a location ID, the location ID must be configured in the respective destinations.
5. (Optional) You can provide a `<Description>` of the subaccount that is shown when clicking on the *Details* icon in the *Actions* column.
6. Choose *Save*.

Next Steps

- To modify an existing subaccount, choose the *Edit* icon and change the `<Display Name>`, `<Location ID>` and/or `<Description>`.



- You can also delete a subaccount from the list of connections.The subaccount will be disconnected and all configurations will be removed from the installation.

Related Information

Managing Member Authorizations in the Neo Environment [page 1301]
Copy a Subaccount Configuration [page 294]
Update the Certificate for a Subaccount [page 296]
Configure a Disaster Recovery Subaccount [page 297]
Find Your Subaccount ID (Cloud Foundry Environment) [page 299]
Configure Custom Regions [page 300]

1.4.5.2.2.1 Copy a Subaccount Configuration

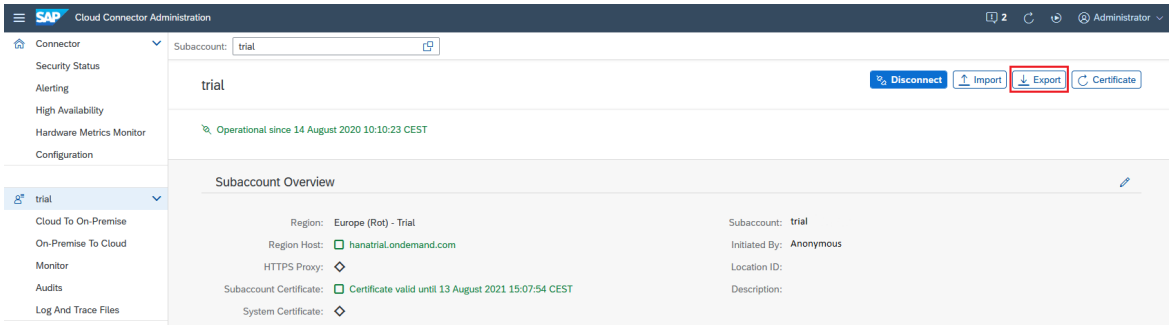
Copy an existing subcaccount configuration in the Cloud Connector to another subaccount.

You can copy the configuration of a subaccount's *Cloud To On-Premise* and *On-Premise To Coud* sections to a new subaccount, by using the export and import functions in the Cloud Connector administration UI.

> **i Note**
>
> Principal propagation configuration (section *Cloud To On-Premise*) is not exported or imported, since it contains subaccount-specific data.

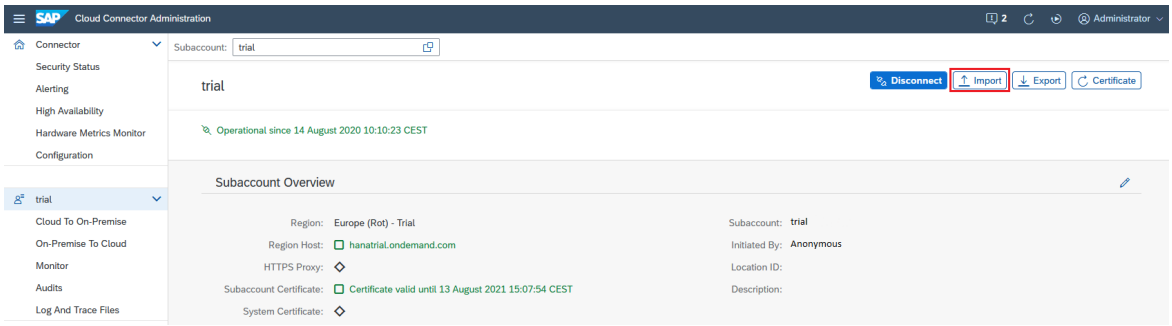
Procedure: Export an Existing Configuration

1. In the Cloud Connector administration UI, choose your subaccount from the navigation menu.
2. To export the existing configuration, choose the *Export* button in the upper right corner. The configuration is downloadad as a zip file to your local file system.

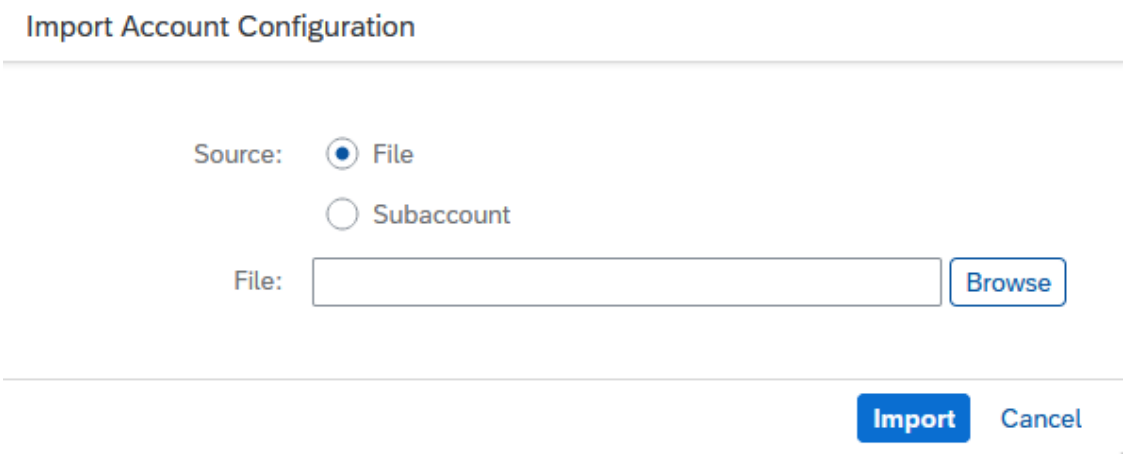


Procedure: Import an Existing Configuration

1. From the navigation menu, choose the subaccount to which you want to copy an existing configuration.
2. To import an existing configuration, choose the *Import* button in the upper right corner.



3. Select one of the following sources:
   1. *File*, if you want to copy the configuration from a previously downloaded zip file.
   2. *Subaccount*, if you want to copy the configuration directly from another existing subaccount.



1.4.5.2.2.2 Update the Certificate for a Subaccount

Certificates used by the Cloud Connector are issued with a limited validity period. To prevent a downtime while refreshing the certificate, you can update it for your subaccount directly from the administration UI.

Prerequisites

You must have the required subaccount authorizations on SAP BTP to update certificates for your subaccount.

See:

- Connectivity: Technical Roles (**Cloud Foundry** environment)
- Connectivity: User Roles (**Neo** environment)

Procedure

> → Tip
>
> You can use this procedure even if the certificate has already expired.

Proceed as follows to update your subaccount certificate:

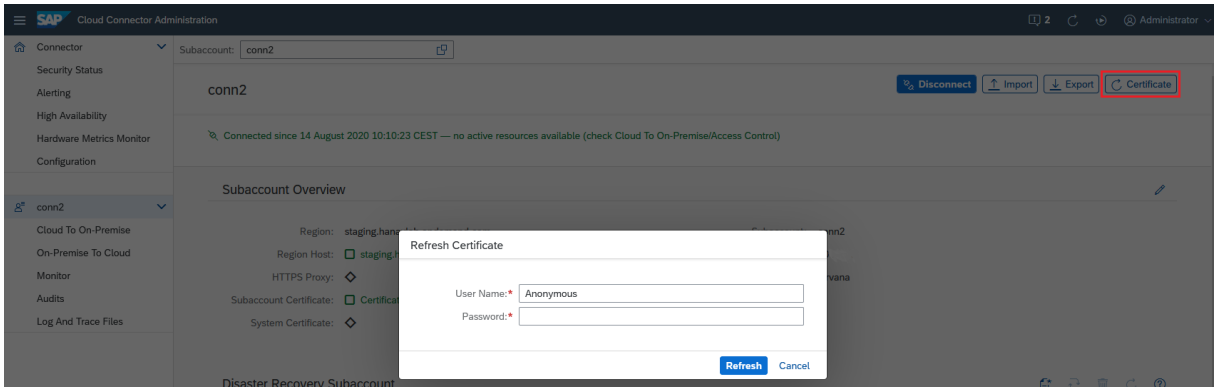
1. From the main menu, choose your subaccount.

   > i Note
   >
   > To check the certificate's validity, click on the `<Subaccount Certificate>` in section *Subaccount Overview*.

2. Choose the *Certificate* button. A dialog opens, requesting a user name and password.
3. Enter `<User Name>` and `<Password>` and choose *OK*. The certificate assigned to your subaccount is refreshed.

   > i Note
   >
   > In the **Neo** environment, the user must have the role `Cloud Connector Admin` or `Administrator`. For `<User Name>`, provide your **`Login E-mail`** or your user ID.

4. If you have configured a disaster recovery subaccount, go to section *Disaster Recovery Subaccount* below and choose *Refresh Disaster Recovery Certificate*.
5. Enter `<User Name>` and `<Password>` as in step 3 and choose *OK*.



1.4.5.2.2.3 Configure a Disaster Recovery Subaccount

Configure a subaccount as backup for disaster recovery.

Each subaccount (except trial accounts) can optionally have a disaster recovery subaccount.

Prerequisite is that you are using the enhanced disaster revovery, see What is Enhanced Disaster Recovery.

The disaster recovery subaccount is intended to take over if the region host of its associated original subaccount faces severe issues.

A disaster recovery account inherits the configuration from its original subaccount except for the region host. The user can, but does not have to be the same.

Procedure

1. From the main menu, choose your subaccount.
2. In section *Disaster Recovery Subaccount*, choose *Configure disaster recovery subaccount*.
3. In the configuration dialog, select an appropriate `<Region Host>` from the drop-down list.

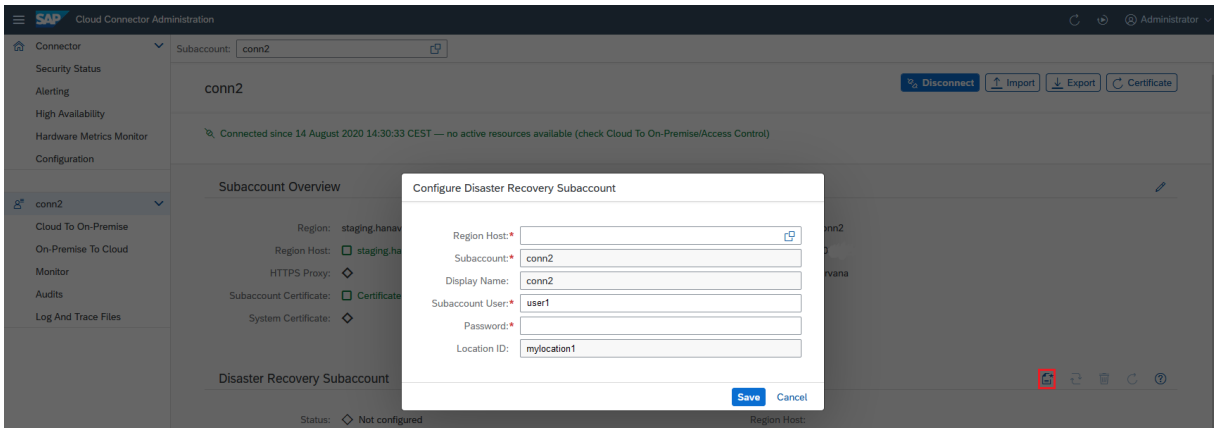
   > **i Note**
   >
   > The selected region host must be different from the region host of the original subaccount.

4. (Optional) You can adjust the `<Subaccount User>`.
5. Enter the `<Password>` for the subaccount user.
6. If configured, enter a `<Location ID>`.
7. Choose *Save*.

> **i Note**
>
> The technical subaccount name, the display name, and the location ID must remain the same. They are set automatically and cannot be changed.

> ### Note
>
> You cannot choose another original subaccount nor a trial subaccount to become a disaster recovery subaccount.

> ### Note
>
> If you want to change a disaster recovery subaccount, you must delete it first and then configure it again.



To switch from the original subaccount to the disaster recovery subaccount, choose *Employ disaster recovery subaccount*.

The disaster recovery subaccount then becomes active, and the original subaccount is deactivated.

You can switch back to the original subaccount as soon as it is available again.

> ### Note
>
> As of Cloud Connector 2.11, the cloud side informs about a disaster by issuing an event. In this case, the switch is performed automatically.

Related Information

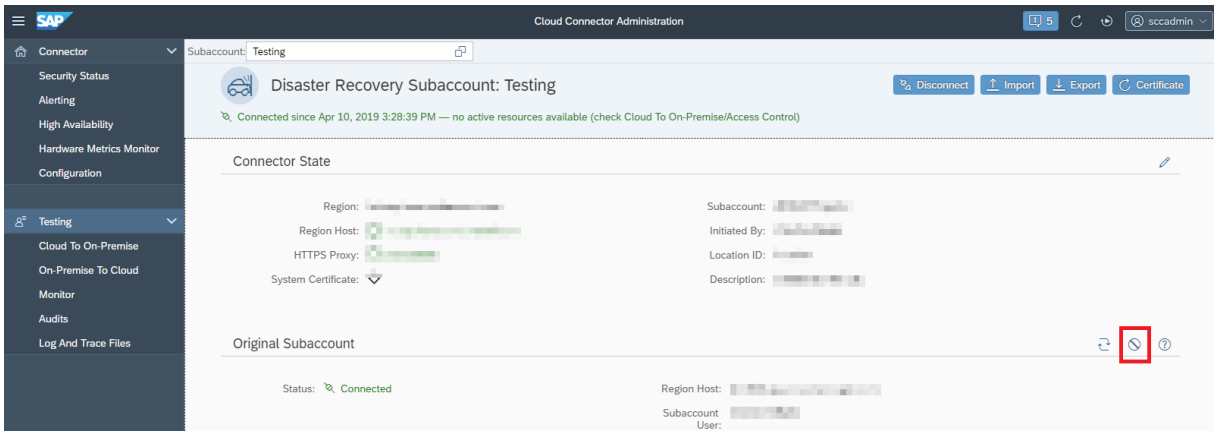
Convert a Disaster Recovery Subaccount into a Standard Subaccount [page 298]

1.4.5.2.2.3.1 Convert a Disaster Recovery Subaccount into a Standard Subaccount

Convert a disaster recovery sucaccount into a standard subaccount if the former primary subaccount's region cannot be recovered.

Disaster recovery subaccounts that were switched to disaster recovery mode can be elevated to standard subaccounts if a disaster recovery region replaces an original region that is not expected to recover.

If a disaster recovery subaccount should be used as primary subaccount, you can convert it by choosing the button *Discard original subaccount and replace it with disaster recovery subaccount*.

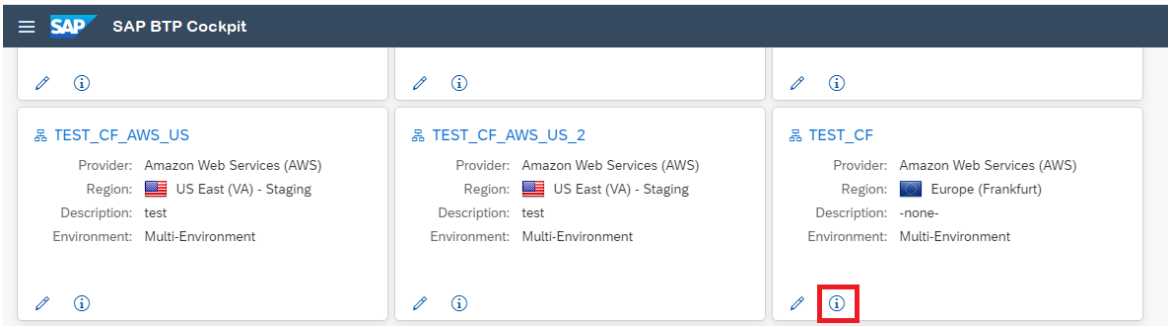


1.4.5.2.2.4 Find Your Subaccount ID (Cloud Foundry Environment)

Get your subaccount ID to configure the Cloud Connector in the Cloud Foundry environment.

In order to set up your subaccount in the Cloud Connector, you must know the subaccount ID. Follow these steps to acquire it:

1. Open the SAP BTP cockpit.
2. Navigate to the subaccount list of the global account containing your subaccount: choose *Home* > *<Your Global Account>* > *Account Explorer*.
3. Find your subaccount in the list.
4. Choose the *Info* icon in the subaccount tile to display the subaccount ID:



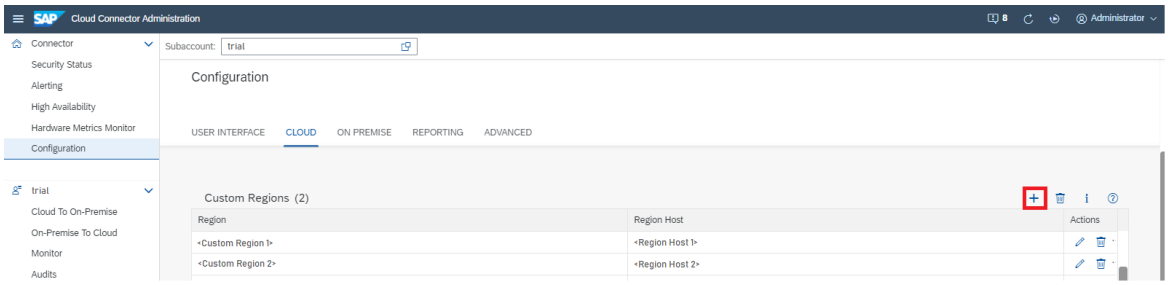
1.4.5.2.2.5 Configure Custom Regions

Configure regions that are not available in the standard selection.

If you want to use a custom region for your subaccount, you can configure regions in the Cloud Connector, which are not listed in the selection of standard regions.

To add a custom region, do the following:

1. From the Cloud Connector main menu, choose *Configuration* > *Cloud* and go to the *Custom Regions* section.
2. To add a region to the list, choose the *Add* icon.



3. In the *Add Region* dialog, enter the `<Region>` and `<Region Host>` you want to use.
4. Choose *Save*.
5. To edit a region from the list, select the corresponding line and choose the *Edit* icon.

1.4.5.2.3 Authenticating Users against On-Premise Systems

Authentication types supported by the Cloud Connector.

Currently, the Cloud Connector supports **basic authentication** and **principal propagation** (user propagation) as user authentication types towards internal systems. The destination configuration of the used cloud application defines which of these types is used for the actual communication to an on-premise system through the Cloud Connector, see Managing Destinations.

- To use **basic authentication**, configure an on-premise system to accept basic authentication and to provide one or multiple service users. No additional steps are necessary in the Cloud Connector for this authentication type.
- To use **principal propagation**, you must explicitly configure trust to those cloud entities from which user tokens are accepted as valid. You can do this in the *Trust* view of the Cloud Connector, see Set Up Trust for Principal Propagation [page 301].

Related Information

Configuring Principal Propagation [page 301]

1.4.5.2.3.1 Configuring Principal Propagation

Use principal propagation to simplify the access of SAP BTP users to on-premise systems.

Tasks in this section:

Task	Description
Set Up Trust for Principal Propagation [page 301]	Configure a trusted relationship in the Cloud Connector to support principal propagation. Principal propagation lets you forward the logged-on identity in the cloud to the internal system without requesting a password.
Configure a CA Certificate for Principal Propagation [page 304]	Install and configure an X.509 certificate to enable support for principal propagation.
Configuring Principal Propagation to an ABAP System [page 308]	Learn more about the different types of configuring and supporting principal propagation for a particular AS ABAP.
Configure a Subject Pattern for Principal Propagation [page 320]	Define a pattern identifying the user for the subject of the generated short-lived `X.509` certificate, as well as its validity period.
Configure a Secure Login Server [page 322]	Configuration steps for Java Secure Login Server (SLS) support.
Configure Kerberos [page 326]	The Cloud Connector lets you propagate users authenticated in SAP BTP via Kerberos against back-end systems. It uses the **Service For User and Constrained Delegation** protocol extension of Kerberos.
Configuring Principal Propagation to SAP NetWeaver AS for Java [page 328]	Find step-by-step instructions on how to configure principal propagation to an application server Java (AS Java).

Related Information

Principal Propagation (Cloud Foundry environment)

Principal Propagation [page 129] (Neo environment)

1.4.5.2.3.1.1 Set Up Trust for Principal Propagation

Establish trust to an identiy provider to support principal propagation.

Tasks

Configure Trusted Entities in the Cloud Connector [page 302]

Configure an On-Premise System for Principal Propagation [page 303]

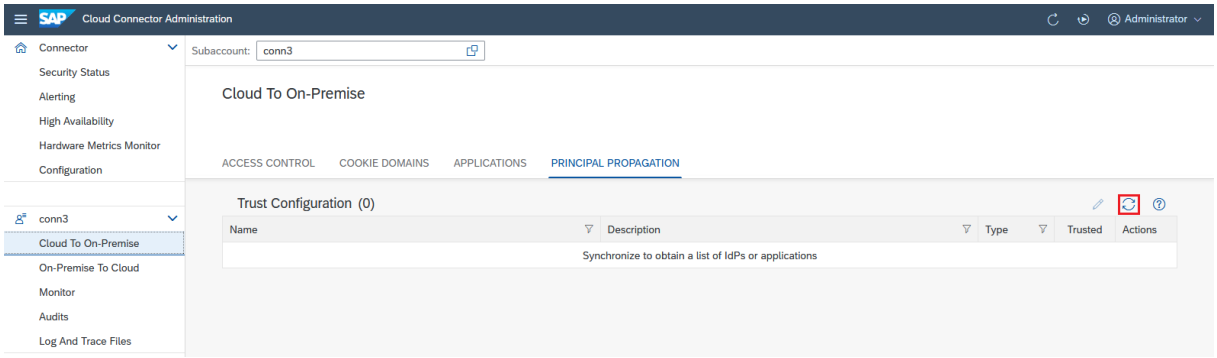
Trust Cloud Applications in the Cloud Connector [page 304]

Set up a Trust Store [page 304]

Configure Trusted Entities in the Cloud Connector

You perform trust configuration to support principal propagation. By default, your Cloud Connector does not trust any entity that issues tokens for principal propagation. Therefore, the list of trusted identity providers is empty by default. If you decide to use the principal propagation feature, you must establish trust to at least one identiy provider. Currently, SAML2 identity providers are supported. You can configure trust to one or more SAML2 IdPs per subaccount. After you've configured trust in the cockpit for your subaccount, for example, to your own company's identity provider(s), you can synchronize this list with your Cloud Connector.

As of Cloud Connector `2.4`, you can also trust HANA instances and Java applications to act as identity providers.



From your subaccount menu, choose *Cloud to On-Premise* and go to the *Principal Propagation* tab. Choose the *Synchronize* button to store the list of existing identity providers locally in your Cloud Connector.

Select an entry to see its details:

- `Name`: the name associated with the identity provider.
- `Description`: descriptive information about this entry.
- `Type`: type of the trusted entity.
- `Trusted`: indicates whether the entry is trusted for principal propagation.
- `Actions`: Choose the *Show Certificate Information* icon to display detail information for the corresponding entry. The Cloud Connector runtime will use the certificate associated with the entry to verify that the assertion used for principal propagation was issued by a trusted entity.

You can decide for each entry, whether to trust it for the principal propagation use case by choosing *Edit* and (de)selecting the *Trusted* checkbox.

> **Note**
>
> Whenever you update the SAML IdP configuration for a subaccount on cloud side, you must synchronize the trusted entities in theCloud Connector. Otherwise the validation of the forwarded SAML assertion will fail with an exception containing an exception message similar to this: *Caused by: com.sap.engine.lib.xml.signature.SignatureException: Unable to validate signature -> java.security.SignatureException: Signature decryption error: javax.crypto.BadPaddingException: Invalid PKCS#1 padding: encrypted message and modulus lengths do not match!*.

Back to Tasks [page 302]

Configure an On-Premise System for Principal Propagation

Set up principal propagation from SAP BTP to your internal system that is used in a hybrid scenario.

> **Note**
>
> As a prerequisite for principal propagation for RFC, the following cloud application runtime versions are required:
>
> - for Java Web: `1.51.8` or higher
> - for Java EE 6 Web Profile: `2.31.11` or higher
> - other runtimes support it with any version

1. Set up trust to an entity that is issuing an assertion for the logged-on user (see section above).
2. Set up the system identity for the Cloud Connector.
   - For HTTPS, you must import a system certificate into your Cloud Connector.
   - For RFC, you must import an SNC PSE into your Cloud Connector.
3. Configure the target system to trust the Cloud Connector.
   There are two levels of trust:
   1. First, you must allow the Cloud Connector to identify itself with its system certificate (for HTTPS), or with the SNC PSE (for RFC).
   2. Then, you must allow this identity to propagate the user accordingly:
      - For HTTPS, the Cloud Connector forwards the true identity in a short-lived `X.509` certificate in an HTTP header named **`SSL_CLIENT_CERT`**. The system must use this certificate for logging on the real user. The SSL handshake, however, is performed through the system certificate. For more information on identity forwarding, see Configure Access Control (HTTP) [page 331].
      - For RFC, the Cloud Connector forwards the true identity as part of the RFC protocol.

   For more information, see Configuring Principal Propagation to an ABAP System [page 308].
4. Configure the user mapping in the target system. The `X.509` certificate contains information about the cloud user in its subject. Use this information to map the identity to the appropriate user in this system. This step applies for both HTTPS and RFC.

> **i Note**
>
> If you have the following scenario: *Application1->AppToAppSSO->Application2->Principal Propagation->On premise Backend System* you must mark *Application2* as **`trusted`** by the Cloud Connector in tab *Principal Propagation*, section *Trust Configuration*.
>
> If you use an identity provider that issues unsigned assertions, you must mark all relevant applications as **`trusted`** by the Cloud Connector in tab *Principal Propagation*, section *Trust Configuration*.

Back to Tasks [page 302]

Trust Cloud Applications in the Cloud Connector

Configure an allowlist for trusted cloud applications, see Configure Trust [page 452].

Back to Tasks [page 302]

Set up a Trust Store

Configure a trust store that acts as an allowlist for trusted on-premise systems. See Configure Trust [page 452].

Back to Tasks [page 302]

Related Information

Principal Propagation (Cloud Foundry)

Principal Propagation [page 129] (Neo)

1.4.5.2.3.1.2 Configure a CA Certificate for Principal Propagation

Install and configure an X.509 certificate to enable support for principal propagation in the Cloud Connector.

Supported CA Mechanisms

You can enable support for principal propagation with X.509 certificates by performing either of the following procedures:

- Using a *Local CA* in the Cloud Connector.

  > **i Note**
  >
  > Prior to version 2.7.0, this was the only option and the system certificate was acting both as client certificate and CA certificate in the context of principal propagation.

- Using a *Secure Login Server* and delegate the CA functionality to it.

The Cloud Connector uses the configured CA approach to issue short-lived certificates for logging on the same identity in the back end that is logged on in the cloud. For establishing trust with the back end, the respective configuration steps are independent of the approach that you choose for the CA.

Install a local CA Certificate

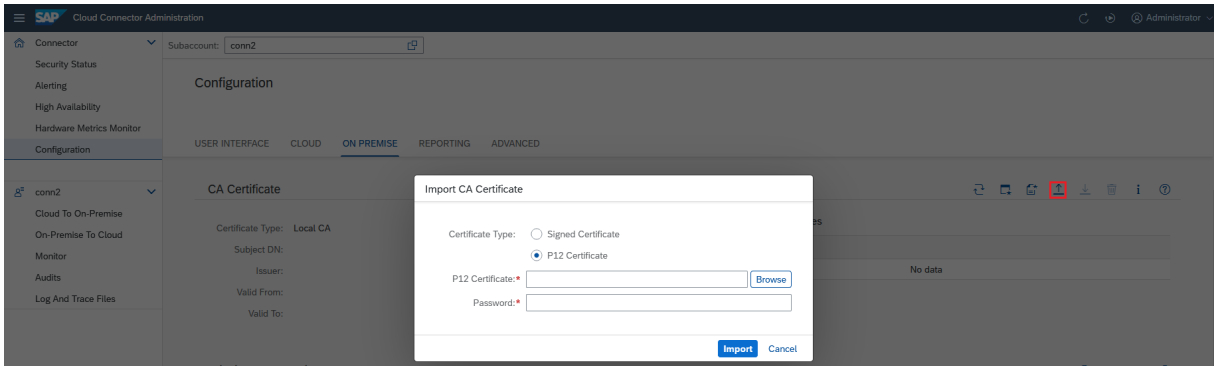
To issue short-lived certificates that are used for principal propagation to a back-end system, you can import an X.509 client certificate into the Cloud Connector. This CA certificate must be provided as *PKCS#12* file containing the (intermediate) certificate, the corresponding private key, and the CA root certificate that signed the intermediate certificate (plus the certificates of any other intermediate CAs, if the certificate chain includes more than those two certificates).

Use either of the following options to install a local CA certificate:

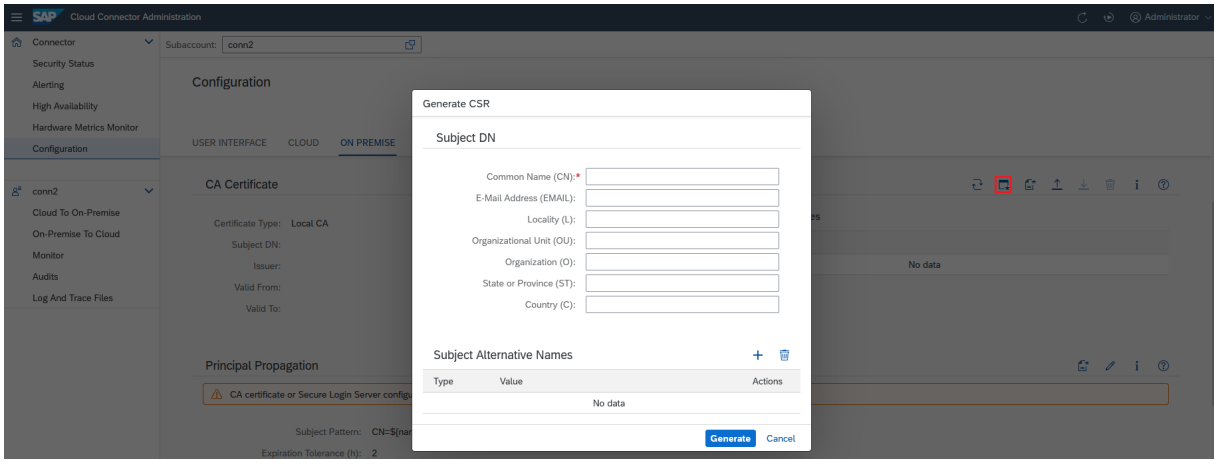
- Option 1: Choose the PKCS#12 file from the file system, using the file upload dialog. For the import process, you must also provide the file password.
- Option 2: Start a *Certificate Signing Request* (CSR) procedure like for the UI certificate, see Recommended: Exchange UI Certificates in the Administration UI [page 270].
- Option 3: (As of version 2.10) Generate a self-signed certificate, which might be useful in a demo setup or if you need a dedicated CA. In particular for this option, it is useful to export the public key of the CA via the button *Download certificate in DER format*.

> **i Note**
>
> The CA certificate should have the `KeyUsage` attribute `keyCertSign`. Many systems verify that the issuer of a certificate includes this attribute and deny a client certificate without this attribute. When using the CSR procedure, the attribute is requested for the CA certificate. Also, when generating a self-signed certificate, this attribute is added automatically.

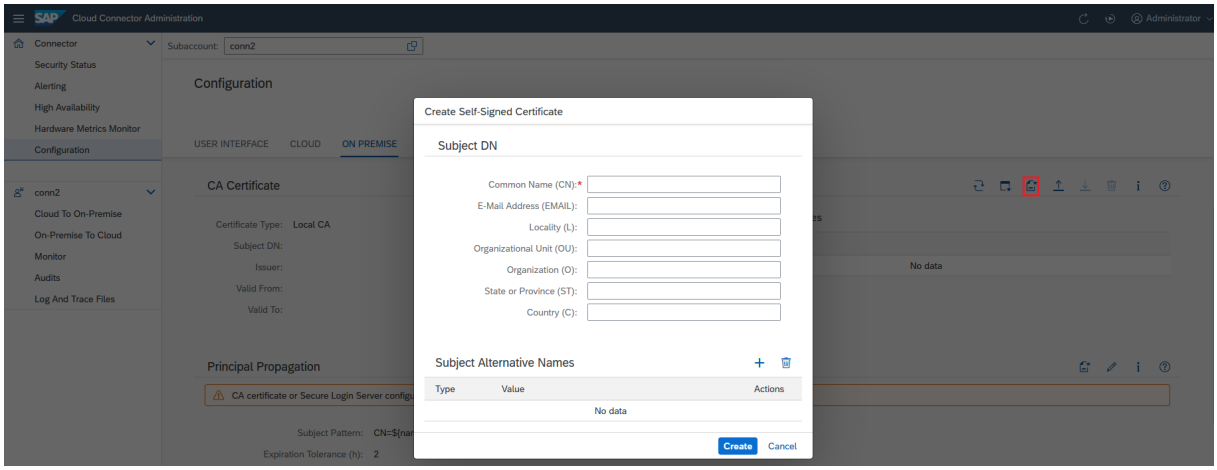
Choose *Import a certificate* for option 1:



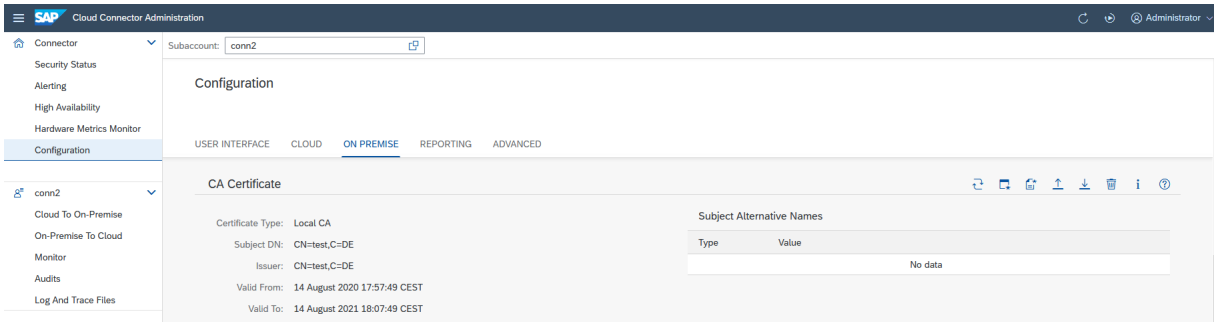
Choose *Generate a certificate signing request* for option 2:



Choose *Create and import a self-signed certificate* if you want to use option 3:



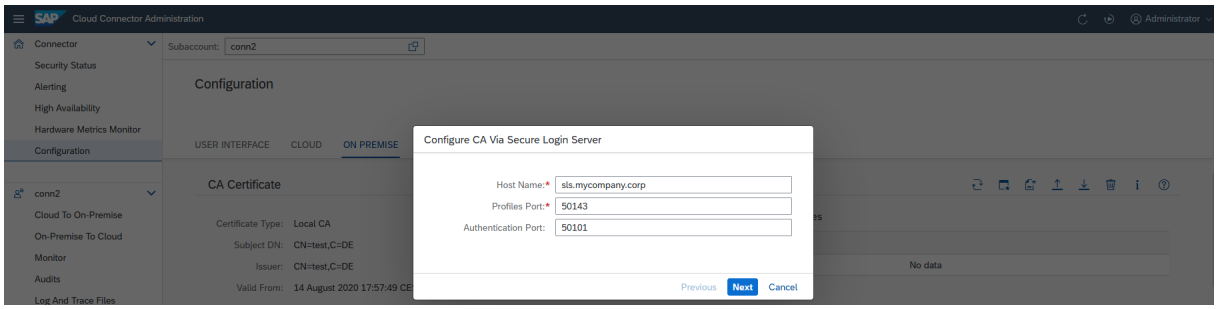
After successful import of the CA certificate, its distinguished name, the name of the issuer, and the validity dates are shown:



If a CA certificate is no longer required, you can delete it. Use the respective *Delete* button and confirm the deletion.

Configure a CA Hosted by a Secure Login Server

If you want to delegate the CA functionality to a Secure Login Server, choose the CA using *Secure Login Server* option and configure the Secure Login Server as follows, after having configured the Secure Login server as described in Configure a Secure Login Server [page 322].



Enter the following:

- `<Host Name>`: The host, on which your Secure Login Server (SLS) is installed.
- `<Profiles Port>`: The profiles port must be provided only when your Secure Login Server is configured to not allow to fetch profiles via the privileged authentication port. In this case, you can provide here the port that is configured for that functionality.
- `<Authentication Port>`: The port, over which the Cloud Connector is requesting the short-lived certificates from SLS. Choose *Next*.

  > **i Note**
  >
  > For this privileged port, a client certificate authentication is required, for which the Cloud Connector system certificate is used.

- `<Profile>`: The Secure Login Server profile that allows to issue certificates as needed for principal propagation with the Cloud Connector.

Choose *Finish* to store the configuration.

Related Information

Configure a Secure Login Server [page 322]
Initial Configuration (HTTP) [page 285]
Initial Configuration (RFC) [page 287]

1.4.5.2.3.1.3 Configuring Principal Propagation to an ABAP System

Learn more about the different types of configuring and supporting principal propagation for a particular AS ABAP.

Task	Description
Configure Principal Propagation for HTTPS [page 308]	Step-by-step instructions to configure principal propagation to an ABAP server for HTTPS.
Configure Principal Propagation via SAP Web Dispatcher [page 312]	Set up a trust chain to use principal propagation to an ABAP server for HTTPS via SAP Web Dispatcher.
Configure Principal Propagation for RFC [page 315]	Step-by-step instructions to configure principal propagation to an ABAP server for RFC.
Rule-Based Mapping of Certificates [page 319]	Map short-lived certificates to users in the ABAP server.

1.4.5.2.3.1.3.1 Configure Principal Propagation for HTTPS

Find step-by-step instructions to configure principal propagation to an ABAP server for HTTPS.

Example Data

The following data are used in this example:

- System certificate was issued by: **`CN=MyCompany CA`**, **`O=Trust Community`**, **`C=DE`**.
- It has the subject: **`CN=SCC`**, **`OU=HCP Scenarios`**, **`O=Trust Community`**, **`C=DE`**.
- The short-lived certificate has the subject **`CN=P1234567890`**, where `P1234567890` is the platform user.

Tasks

Prerequisites [page 309]

Configure an ABAP System to Trust the Cloud Connector's System Certificate [page 309]

Map Short-Lived Certificates to Users [page 311]

Access ICF Services [page 311]

Prerequisites

To perform the following steps, you must have the corresponding authorizations in the ABAP system for the transactions mentioned below (administrator role according to your specific authorization management) as well as an administrator user for the Cloud Connector.

Back to Tasks [page 308]

Back to Example Data [page 308]

1. Configure an ABAP System to Trust the Cloud Connector's System Certificate

This step includes two sub-steps:

Configure the ABAP system to trust the Cloud Connector's system certificate [page 309]

Configure the Internet Communication Manager (ICM) to trust the system certificate for principal propagation [page 310]

Configure the ABAP system to trust the Cloud Connector's system certificate:

1. Open the *Trust Manager* (transaction `STRUST`).
2. Double-click the *SSL-Server Standard* folder in the menu tree on the left.
3. In the displayed screen, click the *Import certificate* button.
4. In the dialog window, choose the certificate file representing the public key of the issuer of the system certificate, for example, in DER format. Typically, this is a CA certificate. If you decide to use a self-signed system certificate, it is the system certificate itself.
5. The details of this certificate are shown in the section above. Using the example certificate data, you would see C**`N=MyCompany CA`**, **`O=Trust Community`**, **`C=DE`** as subject.
6. If you are sure that you are importing the correct certificate, you can integrate the certificate into the certificate list by choosing the *Add to Certificate List* button.
7. Now, the CA certificate (**`CN=MyCompany CA`**, **`O=Trust Community`**, **`C=DE`**) is part of the certificate list.

Back to Step [page 309]

Configure the Internet Communication Manager (ICM) to trust the system certificate for principal propagation:

1. Open the *Profile Editor* (transaction `RZ10`).
2. Select the profile you want to edit, for example, the DEFAULT profile.
3. Select the *Extended maintenance* radio button and choose the *Change* button.
4. Create the following parameter: `icm/trusted_reverse_proxy_<x> = SUBJECT="<subject>", ISSUER="<issuer>"`.
   - Select a free index for **<x>**.
   - **<subject>** is the subject of the system certificate (example data: **CN=SCC**, **OU=BTP Scenarios**, **O=Trust Community**, **C=DE**).
   - **<issuer>** is the issuer of the system certificate (example data: **CN=MyCompany CA**, **O=Trust Community**, **C=DE**).
   - Example: `icm/trusted_reverse_proxy_2 = SUBJECT=" CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE ", ISSUER=" CN=MyCompany CA, O=Trust Community, C=DE "`.

   > **i Note**
   >
   > If your ABAP system uses kernel 7.42 or lower, see SAP note 2052899 or set the following two parameters:
   > - `icm/HTTPS/trust_client_with_issuer`: this is the issuer of the system certificate (example data: **CN=MyCompany CA**, **O=Trust Community**, **C=DE**).
   > - `icm/HTTPS/trust_client_with_subject`: this is the subject of the system certificate (example data: **CN=SCC**, **OU=HCP Scenarios**, **O=Trust Community**, **C=DE**).

   > **⚠ Caution**
   >
   > Make sure that `icm/HTTPS/verify_client` is set to either **1** (request certificate) or **2** (require certificate). If the parameter is set to **0**, trust cannot be established.

5. Save the profile.
6. Open the *ICM Monitor* (transaction `SMICM`) and restart the ICM by choosing *Administration* > *ICM* > *Exit Hard* > *Global*.
7. Verify that the two profile parameters have been taken over by ICM by choosing *Goto* > *Parameters* > *Display*.

> **i Note**
>
> If you have an SAP Web Dispatcher installed in front of the ABAP system, trust must be added in its configuration files with the same parameters as for the ICM. Also, you must add the system certificate of the Cloud Connector to the trust list of the Web dispatcher Server PSE. For more information, see Configure Principal Propagation via SAP Web Dispatcher [page 312].

> **⚠ Caution**
>
> When using principal propagation with X.509 certificates, you cannot use the strict mode in certificate block management (transaction code: CRCONFIG) for the CRL checks within profile *SSL_SERVER*.

Back to Step [page 309]

Back to Tasks [page 308]

Back to Example Data [page 308]

2. Map Short-Lived Certificates to Users

For systems later than SAP NetWeaver 7.3 EHP1 (7.31), you can use rule-based certificate mapping, which is the recommended way to create the required user mappings. For more information, see Rule-Based Mapping of Certificates [page 319].

In older releases (for which this feature does not exist yet), you can do this manually in the system as described below, or use an identity management solution generating the mapping table for a more comfortable approach.

1. Open *Assignment of External ID to Users* (transaction `EXTID_DN`).
2. Switch to the edit mode.
3. Create a new entry. Specify the subject of the certificate as `External ID`. When using the example data, this is **`CN=P1234567890`**. In the *User* field, provide the appropriate ABAP user, for example **`JOHNSMITH`**.
4. Choose *Activate*.
5. Save the mapping.
6. Repeat the previous steps for all users that should be supported for the scenario.

Back to Tasks [page 308]

Back to Example Data [page 308]

3. Access ICF Services

To access the required ICF services for your scenario in the ABAP system, choose one of the following procedures:

- To access ICF services via certificate logon, choose the principal type **`X.509 Certificate (general usage)`** in the corresponding system mapping. This setting lets you use the system certificate for trust as well as for user authentication. For details, see Configure Access Control (HTTP) [page 331], step 7. Additionally, make sure that all required ICF services allow `Logon Through SSL Certificate` as logon method.
- To access ICF services via the logon method `Basic Authentication` (logon with user/password) and principal propagation, choose the principal type **`X.509 Certificate (strict usage)`** in the corresponding system mapping. This setting lets you use the system certificate for trust, but prevents its usage for user authentication. For details, see Configure Access Control (HTTP) [page 331], step 7. Additionally, make sure that all required ICF services allow `Basic Authentication` and `Logon Through SSL Certificate` as logon methods.

- If some of the ICF services require `Basic Authentication`, while others should be accessed via system certificate logon, proceed as follows:
  1. In the Cloud Connector sytem mapping, choose the principal type **`X.509 Certificate (general usage)`** as described above.
  2. In the ABAP system, choose transaction code SICF and go to *Maintain Services*.
  3. Select the service that requires `Basic Authentication` as logon method.
  4. Double-click the service and go to tab *Logon Data*.
  5. Switch to *Alternative Logon Procedure* and ensure that the `Basic Authentication` logon procedure is listed before `Logon Through SSL Certificate`.

> **i** Note
>
> If you are using SAP Web Dispatcher for communication, you must configure it to forward the SSL certficate to the ABAP backend system, see Forward SSL Certificates for X.509 Authentication (SAP Web Dispatcher documentation).

Back to Tasks [page 308]

Back to Example Data [page 308]

Related Information

Configure Principal Propagation via SAP Web Dispatcher [page 312]

Rule-Based Mapping of Certificates [page 319]

Configure a Subject Pattern for Principal Propagation [page 320]

Set Up Trust for Principal Propagation [page 301]

Principal Propagation (Cloud Foundry environment)

Principal Propagation [page 129] (Neo environment)

1.4.5.2.3.1.3.1.1 Configure Principal Propagation via SAP Web Dispatcher

Set up a trust chain to use principal propagation to an ABAP System for HTTPS via SAP Web Dispatcher.

Concept

If you are using an intermediate SAP Web Dispatcher to connect to your ABAP backend system, you must set up a trust chain between the involved components Cloud Connector, SAP Web Dispatcher, and ABAP backend system.

Before configuring the ABAP system (see Configure Principal Propagation for HTTPS [page 308]), in a first step you must configure SAP Web Dispatcher to accept and forward user principals propagated from a cloud account to an ABAP backend.

To do this, follow the step-by-step instructions below.

Example Data

The following data and setup is used for this scenario:

- System certificate was issued by `CN=MyCompany CA, O=Trust Community, C=DE`
- It has the subject `CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE`

Tasks

- Prerequisites [page 313]
- Configure the SAP Web Dispatcher to Trust the Cloud Connector's System Certificate [page 314]
- Next Steps [page 315]

Prerequisites

- Your SAP Web Dispatcher version is 7.53 or higher. See SAP note 908097 for information on recommended SAP Web Dispatcher versions.
- Make sure your SAP Web Dispatcher supports SSL. See Configure SAP Web Dispatcher to Support SSL.
- Ensure that SSL client certificates can be used for authentication in the backend system. See How to Configure SAP Web Dispatcher to Forward SSL Certificates for X.509 Authentication for step-by-step instructions.

Back to Tasks [page 313]

Back to Concept [page 312]

Configure SAP Web Dispatcher to Trust the Cloud Connector's System Certificate

To allow Cloud Connector client certificates for authentication in the backend system, perform the following two steps:

1. **Configure SAP Web Dispatcher to trust the Cloud Connector's system certificate:**
    1. To import the system certificate to SAP Web Dispatcher, open the SAP Web Dispatcher administration interface in your browser.

        > **i Note**
        >
        > The interface is usually configured on `/sap/wdisp/admin`.

    2. In the menu, navigate to *SSL and Trust Configuration* and select *PSE Management*.
    3. In the **Manage PSE** section, select **`SAPSSLS.pse`** from the drop-down list. By default, SAPSSLS.pse contains the server certificate and the list of trusted clients that SAP Web Dispatcher trusts as a server.
    4. In the **Trusted Certificates** section, choose *Import Certificate*.
    5. Enter the certificate as *base64-encoded* into the text box. The procedure to export your certificate in such a format is described in Forward SSL Certificates for X.509 Authentication, step 1.

        > **i Note**
        >
        > Typically, this is a CA certificate. If you are using a self-signed system certificate, it's the system certificate itself.

    6. Choose *Import*.
    7. The certificate details are now shown in section **Trusted Certificates**.
2. **Configure SAP Web Dispatcher to trust the Cloud Connector's system certificate for principal propagation:**
    - Create or edit the following parameter in SAP Web Dispatcher:
      `icm/trusted_reverse_proxy_<x> = SUBJECT="<subject>", ISSUER="<issuer>"`
        - Select a free index for `<x>`.
        - `<subject>`: Subject of the system certificate (example data: `CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE`)
        - `<issuer>`: Issuer of the system certificate (example data: `CN=MyCompany CA, O=Trust Community, C=DE`)

      Example: `icm/trusted_reverse_proxy_0 = SUBJECT="CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE", ISSUER="CN=MyCompany CA, O=Trust Community, C=DE"`
    - **[Deprecated]** Create or edit the following two parameters in SAP Web Dispatcher:

      > **i Note**
      >
      > Use the parameters below (instead of `icm/trusted_reverse_proxy_<x>`) only if your kernel release does not yet support parameter `icm/trusted_reverse_proxy_<x>`.

        - `icm/HTTPS/trust_client_with_issuer`: Issuer of the system certificate (example data: `CN=MyCompany CA, O=Trust Community, C=DE`)

- `icm/HTTPS/trust_client_with_subject`: Subject of the system certificate (example data: `CN=SCC, OU=BTP Scenarios, O=Trust Community, C=DE`)

> **Note**
>
> Make sure `icm/HTTPS/verify_client` is set to **`1`** (request certificate) or **`2`** (require certificate). If set to **`0`**, trust cannot be established. The default value is **`1`**, so it is OK if the parameter is not set at all.

Back to Tasks [page 313]

Back to Concept [page 312]

Next Steps

Now you can proceed with:

- Step 1 of the basic principal propagation setup for HTTPS, see Configure an ABAP System to Trust the Cloud Connector's System Certificate [page 309]. However, when using SAP Web Dispatcher, the ABAP backend must trust the *SAP Web Dispatcher* instead of the *Cloud Connector*, see Forward SSL Certificates for X.509 Authentication, step 2 for details.

Then perform the remaining steps of the basic principal propagation setup for HTTPS as described here:

- Map Short-Lived Certificates to Users [page 311]
- Access ICF Services [page 311]

Back to Tasks [page 313]

Back to Concept [page 312]

1.4.5.2.3.1.3.2 Configure Principal Propagation for RFC

Find step-by-step instructions to configure principal propagation to an ABAP server for RFC.

Configuring principal propagation for RFC requires an SNC (Secure Network Communications) connection. To enable SNC, you must configure the ABAP system and the Cloud Connector accordingly.

The following example provides step-by-step instructions for the SNC setup.

> **Note**
>
> It is important that you use the same SNC implementation on both communication sides. Contact the vendor of your SNC solution to check the compatibility rules.

Example Data

The following data and setup is used:

> **i Note**
>
> The parameters provided in this example are based on an SNC implementation that uses the **SAP Cryptographic Library**. Other vendors' libraries may require different values.

- An SNC identity has been generated and installed on the Cloud Connector host. Generating this identity for the SAP Cryptographic Library is typically done using the tool `SAPGENPSE`. For more information, see Configuring SNC for SAPCRYPTOLIB Using SAPGENPSE.
- The ABAP system is configured properly for SNC.

  > **i Note**
  >
  > For the latest system releases, you can use the SSO wizard to configure SNC (transaction code: SNCWIZARD). System prerequisites are described in SAP note 2015966 .

- The Cloud Connector system identity's SNC name is **`p:CN=SCC, OU=SAP CP Scenarios, O=Trust Community, C=DE`**.
- The ABAP system's SNC identity name is **`p:CN=SID, O=Trust Community, C=DE`**. This value can typically be found in the ABAP system instance profile parameter `snc/identity/as` and hence is provided per application server.
- When using the SAP Cryptographic Library, the ABAP system's SNC identity and the Cloud Connector's system identity should be signed by the same CA for mutual authentication.
- The example short-lived certificate has the subject **`CN=P1234567`**, where **`P1234567`** is the SAP BTP application user.

Tasks

1. Configure the ABAP System [page 316]
2. Map Short-Lived Certificates to Users [page 317]
3. Configure the Cloud Connector [page 317]

1. Configure the ABAP System to Trust the Cloud Connector's System SNC identity

1. Open the *SNC Access Control List for Systems* (transaction `SNC0`).
2. As the Cloud Connector does not have a system ID, use an arbitray value for `<System ID>` and enter it together with its SNC name: **`p:CN=SCC, OU=SAP CP Scenarios, O=Trust Community, C=DE`**.
3. Save the entry and choose the *Details* button.

4. In the next screen, activate the checkboxes for *Entry for RFC activated* and *Entry for certificate activated*.
5. Save your settings.

Back to Tasks [page 316]

Back to Example Data [page 316]

2. Map Short-Lived Certificates to Users

You can do this manually in the system as described below or use an identity management solution for a more comfortable approach. For example, for large numbers of users the rule-based certificate mapping is a good way to save time and effort. See Rule-Based Certificate Mapping.

1. Open *Assignment of External ID to Users* (transaction `EXTID_DN`).
2. Switch to the edit mode.
3. Create a new entry. Specify the subject of the certificate as `External ID`. Using the example data, this is CN=P1234567. In the `<User>` field, provide an appropriate ABAP user, for example **JOHNDOE**.
4. Save the mapping.
5. Repeat the previous steps for all users that should be supported for the scenario.

Back to Tasks [page 316]

Back to Example Data [page 316]

3. Configure the Cloud Connector

Prerequisites [page 317]

Set up the Cloud Connector to Use the SNC Implementation [page 318]

Create an RFC Hostname Mapping [page 318]

Prerequisites

- The required security product for the SNC flavor that is used by your ABAP back-end systems, is installed on the Cloud Connector host.
- The Cloud Connector's system SNC identity is associated with the operating system user under which the Cloud Connector process is running.

> **i Note**
>
> SAP note 2642538 provides a description how you can associate an SNC identity of the SAP Cryptographic Library with a user running an external program that uses JCo. If you use the SAP Cryptographic Library as SNC implementation, perform the corresponding steps for the Cloud Connector. When using a different product, contact the SNC library vendor for details.

Back to Step [page 317]

Set up the Cloud Connector to Use the SNC Implementation

1. In the Cloud Connector UI, choose *Configuration* from the main menu, select the *On Premise* tab, and go to the *SNC* section.
2. Provide the fully qualified name of the SNC library (the security product's shared library implementing the GSS API), the SNC name of the above system identity, and the desired quality of protection by choosing the *Edit* icon.
   For more information, see Initial Configuration (RFC) [page 287].

   > **i Note**
   >
   > The example in Initial Configuration (RFC) [page 287] shows the library location if you use the SAP Secure Login Client as your SNC security product. In this case (as well as for some other security products), *SNC My Name* is optional, because the security product automatically uses the identity associated with the current operating system user under which the process is running, so you can leave that field empty. (Otherwise, in this example it should be filled with **`p:CN=SCC, OU=SAP CP Scenarios, O=Trust Community, C=DE`**.)
   >
   > We recommend that you enter **`Maximum Protection`** for `<Quality of Protection>`, if your security solution supports it, as it provides the best protection.

3. Choose *Save* and *Close*.

Back to Step [page 317]

Create an RFC Hostname Mapping

1. In the *Access Control* section of the Cloud Connector, create a hostname mapping corresponding to the cloud-side RFC destination. See Configure Access Control (RFC) [page 338].
2. Make sure you choose **`RFC SNC`** as `<Protocol>` and **`ABAP System`** as `<Back-end Type>`. In the `<SNC Partner Name>` field, enter the ABAP system's SNC identitiy name, for example, **`p:CN=SID, O=Trust Community, C=DE`**.
3. Save your mapping.

Back to Step [page 317]

Back to Tasks [page 316]

Back to Example Data [page 316]

Related Information

Secure Network Communications (SNC)

Using the SAP Cryptographic Library for SNC

Rule-Based Mapping of Certificates [page 319]

Principal Propagation (Cloud Foundry environment)

Principal Propagation [page 129] (Neo environment)

1.4.5.2.3.1.3.3 Rule-Based Mapping of Certificates

Learn how to efficiently map short-lived certificates to users in the ABAP server.

To perform rule-based mapping of certificates in the ABAP server, proceed as follows:

1. Enter a dynamic parameter using transaction RZ11.
    1. Enter the `login/certificate_mapping_rulebased` parameter.
    2. Choose the *Change Value* button.
    3. Enter the value `1`.
    4. Save the value.

    > **i Note**
    >
    > If dynamic parameters are disabled, enter the value using transaction RZ10 and restart the whole ABAP system.

2. Configure rule-based mapping
    1. To create a sample certificate with the Cloud Connector, logon to the Cloud Connector UI and from the main menu, choose *Configuration*. In the *System Certificate* section, enter a sample CN name and download the sample certificate to the *Downloads* folder of your machine.
    2. To import the sample certificate into the ABAP server, choose transaction CERTRULE and select*Import certificate*.

        > **i Note**
        >
        > To access transaction CERTRULE, you need the corresponding authorizations (see: Assign Authorization Objects for Rule-based Mapping [page 320]).

    3. To create explicit rule mappings, choose the *Rule* button.
    4. Choose *Save*.

        > **i Note**
        >
        > When you save the changes and return to transaction CERTRULE, the sample certificate which you imported in Step 2b will not be saved. This is just a sample editor view to see the sample certificates and mappings.

Related Information

Rule-Based Certificate Mapping

1.4.5.2.3.1.3.3.1 Assign Authorization Objects for Rule-based Mapping

Assign authorizations to access transaction CERTRULE.

To access transaction CERTRULE, you need the following authorizations:

- CC control center: System administration (S_RZL_ADM)
  - `Activity 03` grants display authorizations.
  - `Activity 01` grants change authorizations.
- User Master Maintenance: User Groups (S_USER_GRP)
  - `Activity 03` grants display authorizations.
  - `Activity 02` grants change authorizations.
  - `Class`: enter the names of user groups for which the administrator can maintain explicit mappings.

To assign these authorization objects, proceed as follows:

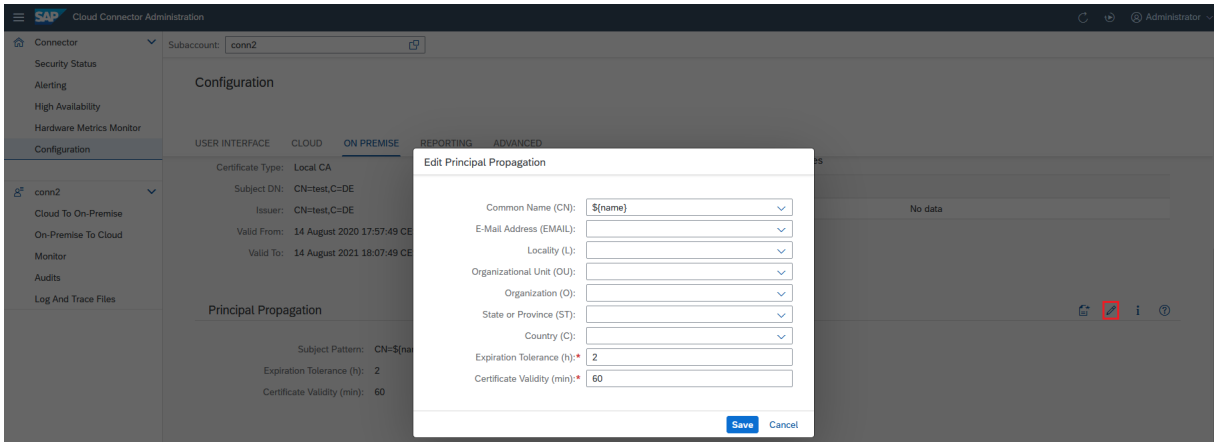
1. Create a *Single Role* using transaction PFCG.
2. To add the authorization objects **`S_RZL_ADM`** and **`S_USER_GRP`**, go to the *Authorizations* tab, choose *Change Authorization data* and select the *Manually* button .
3. To generate the profile, choose *Generate* and save the changes.
4. In the *User* tab, enter the user who should execute the transaction CERTRULE.
5. To match the generated profile to the users, choose *User comparison* .

1.4.5.2.3.1.4 Configure a Subject Pattern for Principal Propagation

Define a pattern identifying the user for the subject of the generated short-lived `X.509` certificate, as well as its validity period.

Configure a Subject Pattern

To configure such a pattern, choose *Configuration* > *On Premise* and press the *Edit* icon in section *Principal Propagation*:



Subject Pattern Details

Use either of the following procedures to define the subject's distinguished name (DN), for which the certificate will be issued:

- Enter the values in the subject pattern fields manually.
- Use the selection menu of the corresponding field to enter a predefined parameter (constant).

Using the selection menu, you can assign values for the following parameters:

- `${name}`
- `${mail}`
- `${display_name}`
- `${login_name}` (as of Cloud Connector version 2.8.1.1)

> **i Note**
>
> If the token provided by the Identity Provider contains additional values that are stored in attributes with different names, but you still want to use it for the subject pattern, you can edit the variable name to place the corresponding attribute value in the subject accordingly. For example, provide **`${email}`**, if a SAML assertion uses `email` instead of providing `mail`.
>
> When using a subaccount in the **Cloud Foundry** environment: As of version 2.12.5, the Cloud Connector also offers direct access to custom variables injected in the JWT (JSON Web token) by SAP BTP *Authorization & Trust Management* that were taken over from a SAML assertion.

The values for these variables are provided by the trusted Identiy Provider in the token which is passed to the Cloud Connector and specifies the user that has logged on to the cloud application.

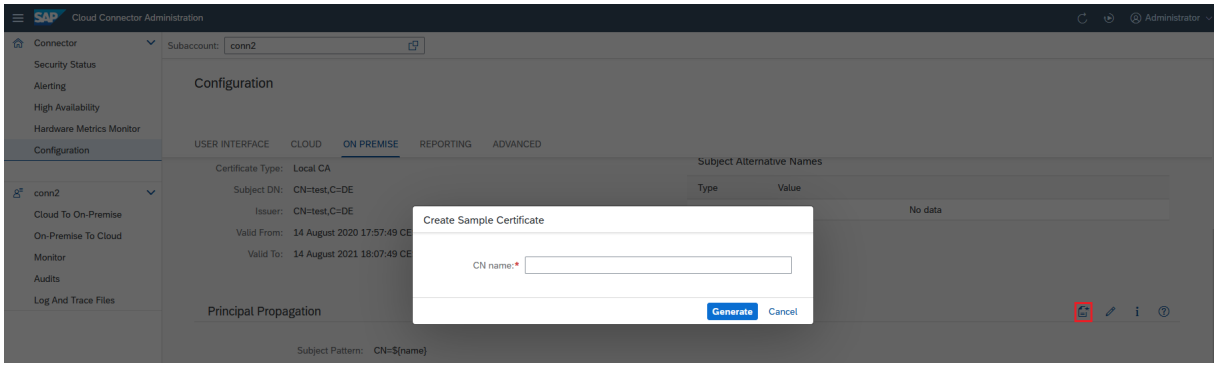
By default, the following attributes are provided:

- `<CN>`: (common name) – the name of the certificate owner
- `<EMAIL>`: (e-mail address) - the e-mail address of the certificate owner
- `<L>`: (locality) – the certificate owner's location
- `<O>`: (organization) – the certificate owner's organization or company

- `<OU>`: (name of organizational unit) – the organizational unit to which the certificate owner belongs
- `<ST>`: (state of residence) – the state in which the certificate issuer resides
- `<C>`: (country of residence) – the country in which the certificate owner resides
- `<Expiration Tolerance (h)>`: The length of time in hours, that an application can use a principal issued for a user after the token from cloud side has expired.
- `<Certificate Validity (min)>`: The length of time in minutes, that a certificate generated for principal propagation can authenticate against the back end. You can reuse a previously generated certificate to improve performance.

Sample Certificate

By choosing *Generate Sample Certificate* you can create a sample certificate that looks like one of the short-lived certificates created at runtime. You can use this certificate to, for example, generate user mapping rules in the target system, via transaction CERTRULE in an ABAP system. If your subject pattern contains variable fields, a wizard lets you provide meaningful values for each of them and eventually you can save the sample certificate in `DER` format.



Related Information

Server Certificate Authentication

1.4.5.2.3.1.5 Configure a Secure Login Server

Configuration steps for Java SLS support.

Content

Overview [page 323]

Requirements [page 324]

Implementation [page 324]

Overview

The Cloud Connector can use on-the-fly generated X.509 user certificates to log in to on-premise systems if the external user session is authenticated (for example by means of SAML). If you do not want to use the built-in certification authority (CA) functionality of the Cloud Connector (for example because of security considerations), you can connect SAP SSO 2.0 Secure Login Server (SLS).

SLS is a Java application running on AS JAVA 7.20 or higher, which provides interfaces for certificate enrollment.

SLS supports the following formats:

- HTTPS
- REST
- JSON
- PKCS#10/PKCS#7

> **i Note**
>
> Any enrollment requires a successful user or client authentication, which can be a single, multiple or even a multi factor authentication.

The following schemes are supported:

- LDAP/ADS
- RADIUS
- SAP SSO OTP
- ABAP RFC
- Kerberos/SPNego
- X.509 TLS Client Authentication

SLS lets you define arbitrary enrollment profiles, each with a unique profile UID in its URL, and with a configurable authentication and certificate generation.

Back to Content [page 323]

Requirements

For user certification, SLS must provide a profile that adheres to the following:

- Cloud Connector client authentication by its X.509 service certificate
- Cloud Connector service certificate and SLS may live in different PKIs
- Cloud Connector hands over the full user´s certificate subject name

With SAP SSO 2.0 SP06, SLS provides the following required features:

- TLS Client Authentication-based enrollment with `SecureLoginModuleUserDelegationWithSSL` (available since SP04)
- multi-PKI support is implemented by all standard components of Application Server (AS) JAVA, AS ABAP, HANA, by importing trusted root CA certificates
- SLS allows `PKCS10:SUBJECT` in a profile´s certificate configuration (SP06)

Back to Content [page 323]

Implementation

INSTALLATION

Follow the standard installation procedures for SLS. This includes the initial setup of a PKI (public key infrastructure).

> **i Note**
>
> SLS allows you to set up one or more own PKIs with Root CA, User CA, and so on. You can also import CAs as PKCS#12 file or use a hardware security module (HSM) as "External User CA".

> **i Note**
>
> You should only use HTTPS connections for any communication with SLS. AS JAVA / ICM supports TLS, and the default configuration comes with a self-signed sever certificate. You may use SLS to replace this certificate by a PKI certificate.

CONFIGURATION

SSL Ports

1. Open the NetWeaver Administrator, choose *Configuration* > *SSL* and define a new port with **`Client Authentication Mode = REQUIRED`**.

   > **i Note**
   >
   > You may also define another port with **`Client Authentication Mode = Do not request`** if you did not do so yet.

2. Import the root CA of the PKI that issued your Cloud Connector service certificate.

3. Save the configuration and restart the Internet Communication Manager (ICM).

Authentication Policy

1. Open the NetWeaver Administrator (NWA, https://<host:port>/nwa).
2. From the top-level menu, choose *Configuration* > *Authentication and Single Sign-On* .
3. In the *Policy Configuration* table, switch to **`Type = Custom`**.
4. Choose *Add* to create a new policy and assign it a name, for example, **`SecureLoginCloudConnector`**.
5. Open *Edit* mode.
6. In the *Details* section of the authentication configuration, choose *Authentication Stack* > *Login Modules* and add **`SecureLoginModuleUserDelegationWithSSL`**.
7. In `<Rule1.subjectName>` and `<Rule1.issuerName>`, enter the respective certificate names of your Cloud Connector service certificate.
8. In the *Details* section of the authentication configuration, choose *Properties* and add the property `UserNameMapping` with value **`VirtualUser`**.
9. Save the policy.

Client Authentication Profile

1. Open the SLS Administration Console (SLAC, https://host:port/slac).
2. From the top-level menu, choose *Profile Management* > *Authentication Profiles* .
3. Create a new profile with `Client Type = Secure Login Client` and assign it a name, for example, **`Cloud Connector User Certificates`**.
4. Choose *User Authentication* > *Use Policy Configuration* and select `Policy Configuration Name = SecureLoginCloudConnector`.
5. Edit all required fields in the wizard according to your requirements.
6. Save your entries.
7. Select the new profile and open *Edit* mode.
8. Choose *Certificate Configuration* > *Certificate Name and Alternative Names* and set `Appendix Subject Name = (PKCS10:SUBJECT)`.
9. Leave all other fields in *Certificate Name and Alternative Names* empty.
10. On the *Enrollment Configuration* page, make sure that the `<Enrollment URL>` has the correct value, otherwise edit and fix it:
    1. full DNS name
    2. port *with* TLS Client Authentication (see port number in NWA SSL Configuration).
11. Save your entries.

User Profile Group

1. Open the SLS Administration Console (SLAC, https://host:port/slac).
2. Go to the top level menu and choose *Profile Management* > *User Profile Groups* .
3. Create a new profile group, make sure the `<Policy URL>` has the correct value:
    1. Full DNS name
    2. Port *without* TLS Client Authentication (see port number in NWA SSL Configuration).
4. In tab *Profiles*, add the profile `Cloud Connector User Certificates`.
5. Save your entries.

Root CA Certificate

1. Open SLS Administration Console (SLAC, https://host:port/slac).
2. Go to the top level menu and choose *Certificate Management*.
3. Select the Root CA certificate you are using in your profile.
4. Choose *Export entry* > *X.509 Certificate* and download the certificate file.

Cloud Connector

Follow the standard installation procedure of the Cloud Connector and configure SLS support:

1. Enter the policy URL that points to the SLS user profile group.
2. Select the profile, for example, `Cloud Connector User Certificates`.
3. Import the Root CA certificate of SLS into the Cloud Connector´s Trust Store.

On-Premise Target Systems

Follow the standard configuration procedure for Cloud Connector support in the corresponding target system and configure SLS support.

To do so, import the Root CA certificate of SLS into the system´s truststore:

- **AS ABAP**: choose transaction STRUST and follow the steps in Maintaining the SSL Server PSE's Certificate List.
- **AS Java**: open the Netweaver Administrator and follow the steps described in Configuring the SSL Key Pair and Trusted X.509 Certificates.

Back to Content [page 323]

1.4.5.2.3.1.6 Configure Kerberos

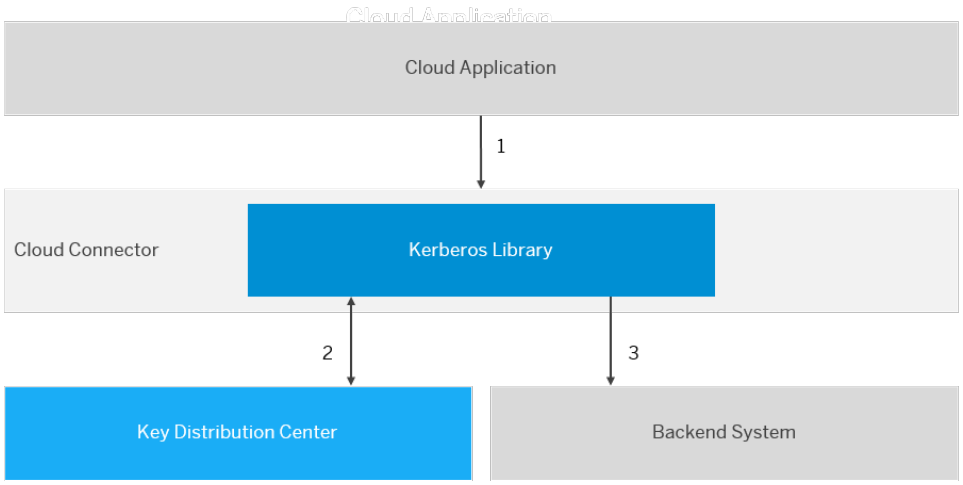
Context

The Cloud Connector allows you to propagate users authenticated in SAP BTP via Kerberos against backend systems. It uses the **Service For User and Constrained Delegation** protocol extension of Kerberos.

> **i** Note
>
> This feature is not supported for ABAP backend systems. In this case, you can use the certificate-based principal propagation, see Configure a CA Certificate for Principal Propagation [page 304].

The Key Distribution Center (KDC) is used for exchanging messages in order to retrieve Kerberos tokens for a certain user and backend system.

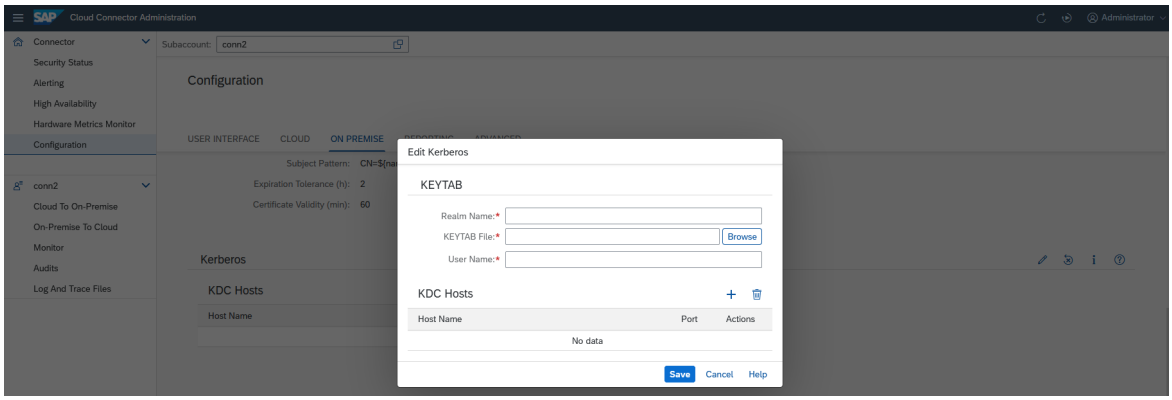
For more information, see Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol .



1. An SAP BTP application calls a backend system via the Cloud Connector.
2. The Cloud Connector calls the KDC to obtain a Kerberos token for the user propagated from the Cloud Connector.
3. The obtained Kerberos token is sent as a credential to the backend system.

Procedure

1. Choose *Configuration* from the main menu.
2. From the *Kerberos* section of the *On Premise* tab, choose *Edit*.



3. Enter the name of your Kerberos realm.
4. Upload a KEYTAB file that contains the secret keys of your service user. The KEYTAB file should contain the **`rc4-hmac`** key for your user.
5. Enter the name of the service user to be used for communication with the KDC. This user should be allowed to request Kerberos tokens for other users for the backend systems that you are going to access.
6. In the `<KDC Hosts>` field (press *Add* to display the field), enter the host name of your KDC using the format **`<host>:<port>`**. The port is optional; if you leave it empty, the default, 88, is used.
7. Choose *Save*.

Example

You have a backend system protected with SPNego authentication in your corporate network. You want to call it from a cloud application while preserving the identity of a cloud-authenticated user.

Define the following:

- A connectivity destination in SAP BTP, with `ProxyType` = **`OnPremise`**.
- A system mapping made in the Cloud Connector. (Choose *Cloud to On Premise* from your subaccount menu, Go to tab *Access Control* > *Add*, and for *Principal Type*, select **`Kerberos`**.)
- Kerberos configuration in the Cloud Connector, where the service user is allowed to delegate calls for your backend host service.

Result:

When you now call a backend system, the Cloud Connector obtains an SPNego token from your KDC for the cloud-authenticated user. This token is sent along with the request to the back end, so that it can authenticate the user and the identity to be preserved.

Related Information

Set Up Trust for Principal Propagation [page 301]

1.4.5.2.3.1.7 Configuring Principal Propagation to SAP NetWeaver AS for Java

Find step-by-step instructions on how to set up an application server for Java (AS Java) to enable principal propagation for HTTPS.

Prerequisites

To perform the following steps, you must have the corresponding administrator authorizations in AS Java (SAP NetWeaver Administrator) as well as an administrator user for the Cloud Connector.

Configure AS Java to Trust the Cloud Connector's System Certificate

Procedure

1. Go to *SAP NetWeaver Administrator* > *Certificates and Keys* and import the Cloud Connector's system certificate into the *Trusted CAs* keystore view. See Importing Certificate and Key From the File System.
2. Configure the Internet Communication Manager (ICM) to trust the system certificate for principal propagation.
   a. Add a new SSL access point. See Adding New SSL Access Points.
   b. Generate a certificate signing request and send it to the CA of your choice. See Configuration of the AS Java Keystore Views for SSL.
   c. Import the certificates and save the configuration.

      Import the certificate signing response, the root X.509 certificate of the trusted CA, and the Cloud Connector's system certificate into the new SSL access point from step 2a. Save the configuration and restart the ICM. See Configuring the SSL Key Pair and Trusted X.509 Certificates.
   d. Test the SSL connection. See Testing the SSL Connection.

Define Rules for User Mapping

Procedure

1. Add the *ClientCertLoginModule* to the policy configuration that the Cloud Connector connects to. See Configuring the Login Module on the AS Java.
2. Define the rules to map users authenticated with their certificate to users that exist in the User Management Engine. See Using Rules for User Mapping in Client Certificate Login Module.
   - To map the user ID of the certificate's subject name field to users, see Using Rules Based on Client Certificate Subject Names.
   - To map the user ID based on rules for the certificate V3 extension *SubjectAlternativeName*, see Using Rules Based on Client Certificate V3 Extensions.
   - To use client certificate filters, see Defining Rules for Filtering Client Certificates.

Related Information

Configuring Transport Layer Security on SAP NetWeaver AS for Java
Using X.509 Client Certificates
Configure Principal Propagation for HTTPS [page 308]

1.4.5.2.4 Configure Access Control

Specify the backend systems that can be accessed by your cloud applications.

To allow your cloud applications to access a certain backend system on the intranet, you must specify this system in the Cloud Connector. The procedure is specific to the protocol that you are using for communication.

Find the detailed configuration steps for each communication protocol here:

Configure Access Control (HTTP) [page 331]

Configure Access Control (RFC) [page 338]

Configure Access Control (LDAP) [page 344]

Configure Access Control (TCP) [page 347]

Copy Access Control Settings

When you add new subaccounts, you can copy the complete access control settings from another subaccount on the same Cloud Connector. You can also do it any time later by using the import/export mechanism provided by the Cloud Connector.

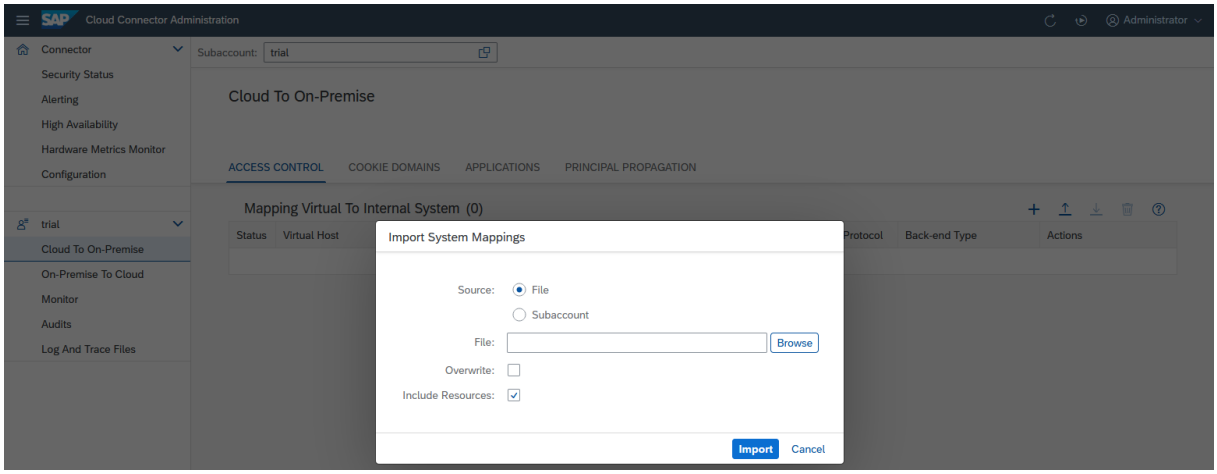
Export Access Control Settings

1. From your subaccount menu, choose *Cloud To On-Premise* and select the tab *Access Control*.
2. To store the current settings in a ZIP file, choose *Download* icon in the upper-right corner.
3. You can later import this file into a different Cloud Connector.

Import Access Control Settings

There are two locations from which you can import access control settings:

- A file that was previously exported from a Cloud Connector
- A different subaccount on the same Cloud Connector



Two additional options define the behavior of the import:

- *Overwrite*: Select this checkbox if you want to replace existing system mappings with imported ones. Do not select this checkbox if you want to keep existing mappings and only import the ones that are not yet available (default).

  > **i Note**
  >
  > A system mapping is uniquely identified by the combination of virtual host and port.

- *Include Resources*: When this checkbox is selected (default), the resources that belong to an imported system are also imported. Otherwise no resources are imported, that is, imported system mappings do not expose any resources.

Related Information

Configure Access Control (HTTP) [page 331]
Configure Access Control (RFC) [page 338]
Configure Access Control (LDAP) [page 344]
Configure Access Control (TCP) [page 347]
Configure Accessible Resources [page 350]
Configure Domain Mappings for Cookies [page 455]

1.4.5.2.4.1 Configure Access Control (HTTP)

Specify the backend systems that can be accessed by your cloud applications using HTTP.

To allow your cloud applications to access a certain backend system on the intranet via HTTP, you must specify this system in the Cloud Connector.

> **i Note**
>
> Make sure that also redirect locations are configured as internal hosts.
>
> If the target server responds with a redirect HTTP status code (30x), the cloud-side HTTP client usually sends the redirect over the Cloud Connector as well. The Cloud Connector runtime then performs a reverse lookup to rewrite the location header that indicates where to route the redirected request.
>
> If the redirect location is ambiguous (that is, several mappings point to the same internal host and port), the first one found is used. If none is found, the location header stays untouched.

Tasks

Expose Intranet Systems [page 332]

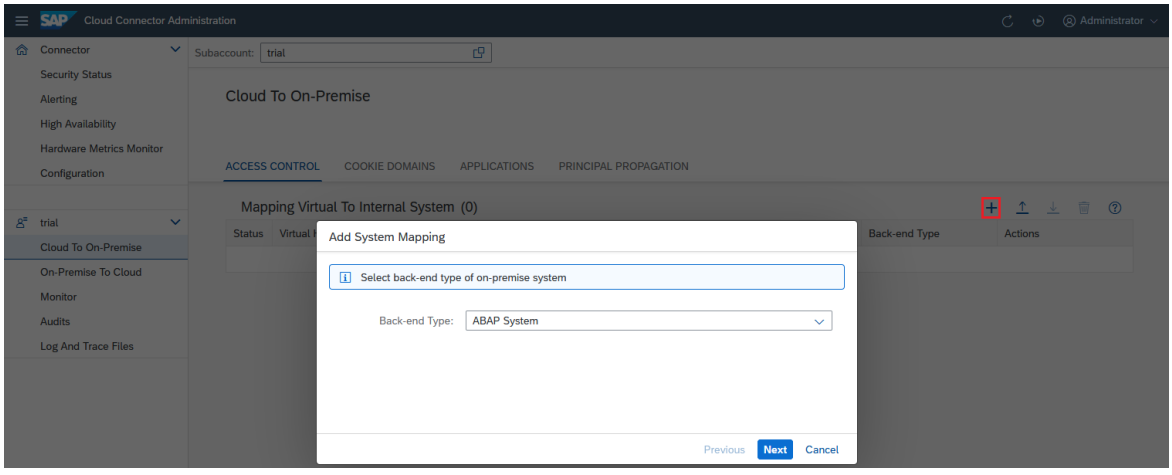
Limit the Accessible Services for HTTP(S) [page 336]

Activate or Suspend Resources [page 337]

Expose Intranet Systems

Insert a new entry in the Cloud Connector access control management:

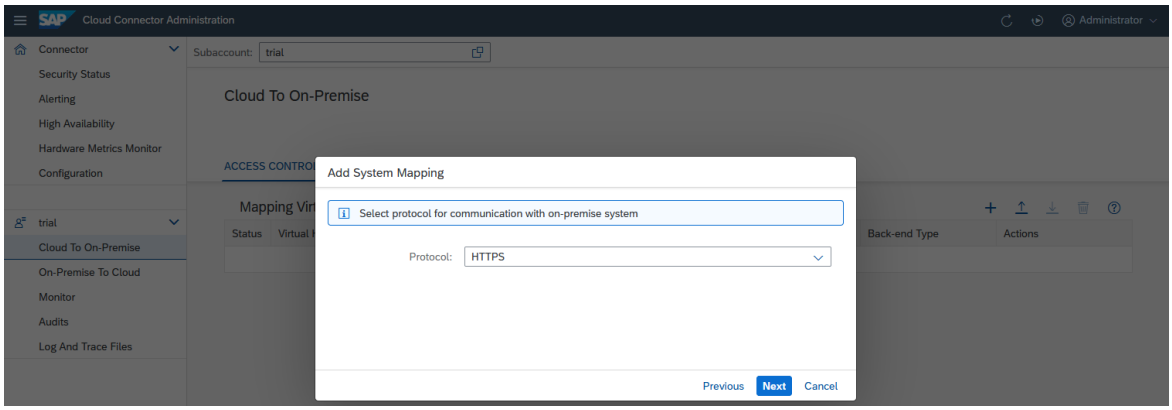
1. Choose *Cloud To On Premise* from your *Subaccount* menu.
2. Choose *Add*. A wizard will open and ask for the required values.
3. *Backend Type*: Select the description that matches best the addressed backend system.
   When you are done, choose *Next*.



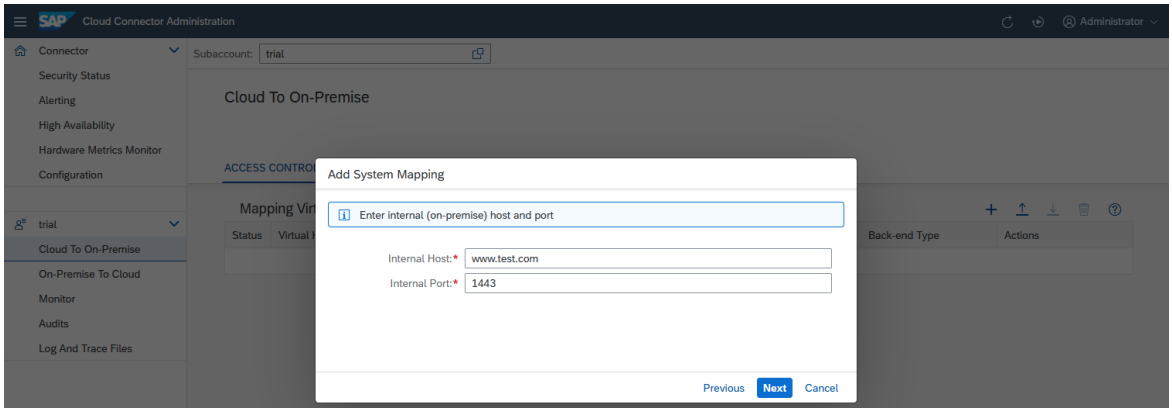
4. *Protocol*: This field allows you to decide whether the Cloud Connector should use **`HTTP`** or **`HTTPS`** for the connection to the backend system. Note that this is completely independent from the setting on cloud side. Thus, even if the HTTP destination on cloud side specifies "`http://`" in its URL, you can select **`HTTPS`**. This way, you are ensured that the entire connection from the cloud application to the actual backend system (provided through the SSL tunnel) is SSL-encrypted. The only prerequisite is that the

backend system supports HTTPS on that port. For more information, see Initial Configuration (HTTP) [page 285].

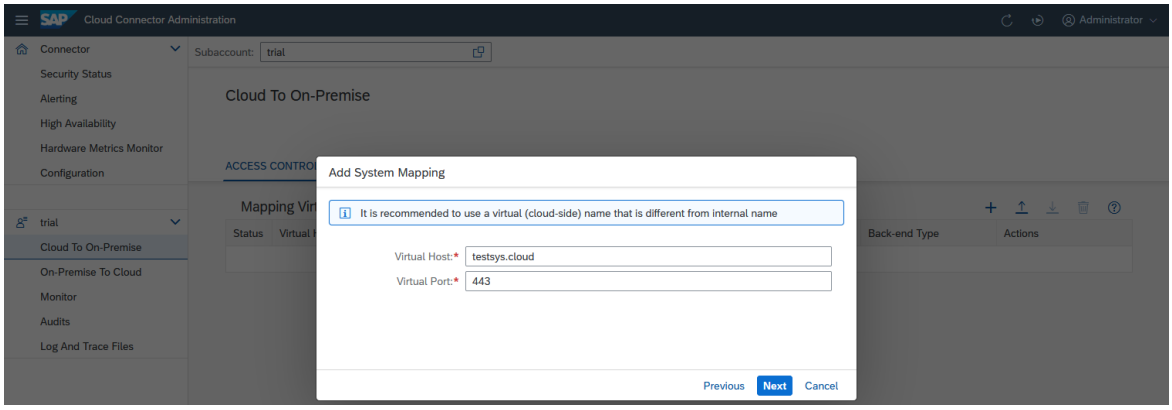
- If you specify HTTPS and there is a "system certificate" imported in the Cloud Connector, the latter attempts to use that certificate for performing a client-certificate-based logon to the backend system.
- If there is no system certificate imported, the Cloud Connector opens an HTTPS connection without client certificate.



5. *Internal Host* and *Internal Port* specify the actual host and port under which the target system can be reached within the intranet. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector without any proxy. Cloud Connector will try to forward the request to the network address specified by the internal host and port, so this address needs to be real.



6. *Virtual Host* specifies the host name exactly as it is specified as the URL property in the HTTP destination configuration in SAP BTP
   See:
   Create HTTP Destinations [page 80] (Neo environment)
   The virtual host can be a fake name and does not need to exist. The *Virtual Port* allows you to distinguish between different entry points of your backend system, for example, **`HTTP/80`** and **`HTTPS/443`**, and have different sets of access control settings for them. For example, some noncritical resources may be accessed by HTTP, while some other critical resources are to be called using HTTPS only. The fields will be prepopulated with the values of the *Internal Host* and *Internal Port*. In case you don't modify them, you must provide your internal host and port also in the cloud-side destination configuration or in the URL used for your favorite HTTP client.

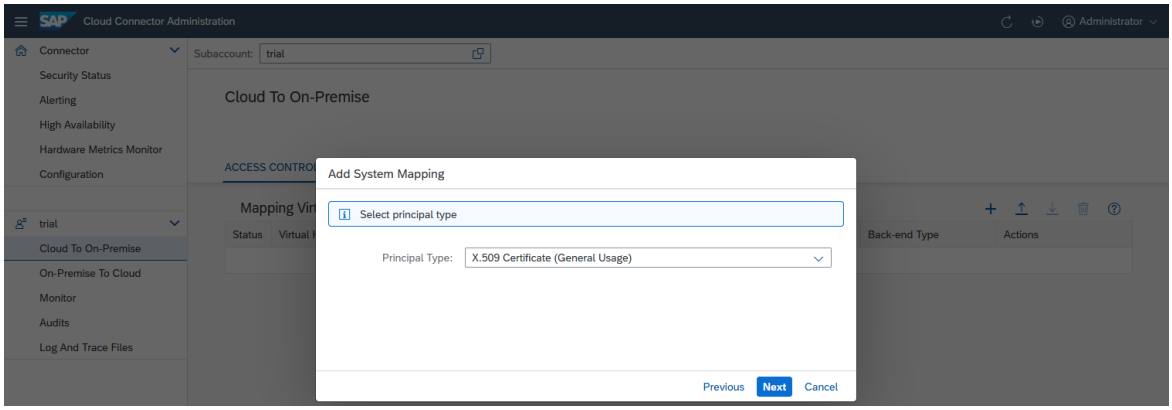


7. *Principal Type*: You can use two different variants of an X.509 certificate to define the principal type that is sent to the backend within the HTTP request: **`X.509 Certificate (General Usage)`** or **`X.509 Certificate (Strict Usage)`**. The latter was introduced with Cloud Connector 2.11.
   If a principal is sent from the cloud side, it is injected in both cases as an HTTP header (`SSL_CLIENT_CERT`) and forwarded to the backend. If the backend is configured correctly for principal propagation, this certificate can be used for authentication.
   However, if the cloud side does not send a principal, the variants behave differently:
   - *General Usage* (as well as `<Principal Type>` = **`None`**) allows to alternatively use the *system* certificate for the *TLS handshake* (actually used for trust) also for authentication.
   - *Strict Usage* does not allow this. In this case, another authentication type (specified in an additional header) is used instead, for example, basic authentication.

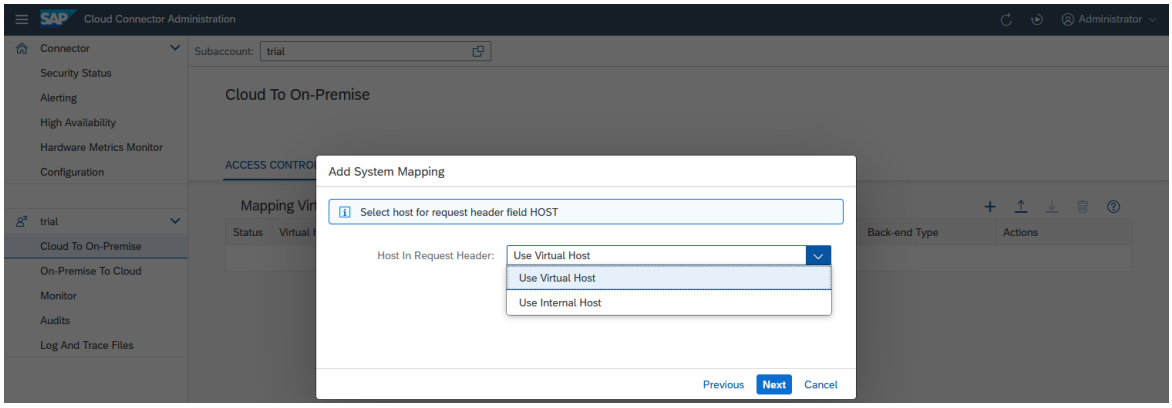
   This setting also applies to HTTP authentication types other than principal propagation.

> **i Note**
>
> The recommended variant is **`X.509 Certificate (Strict Usage)`** as this lets you use principal propagation and, for example, basic authentication over the same access control entry, regardless of the logon order settings in the target system.

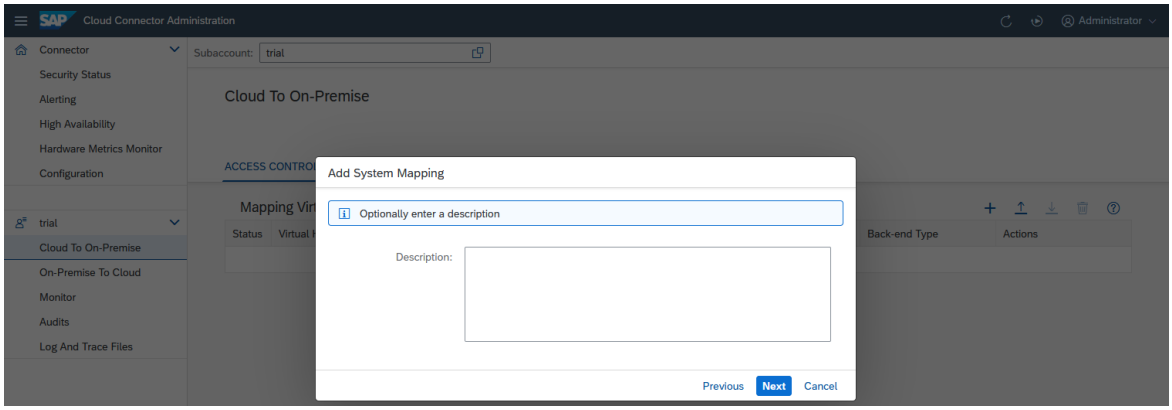
   To prevent the use of principal propagation to the target system, choose **`None`** as `<Principal Type>`. In this case, no principal is injected.
   For more information on principal propagation, see Configuring Principal Propagation [page 301].



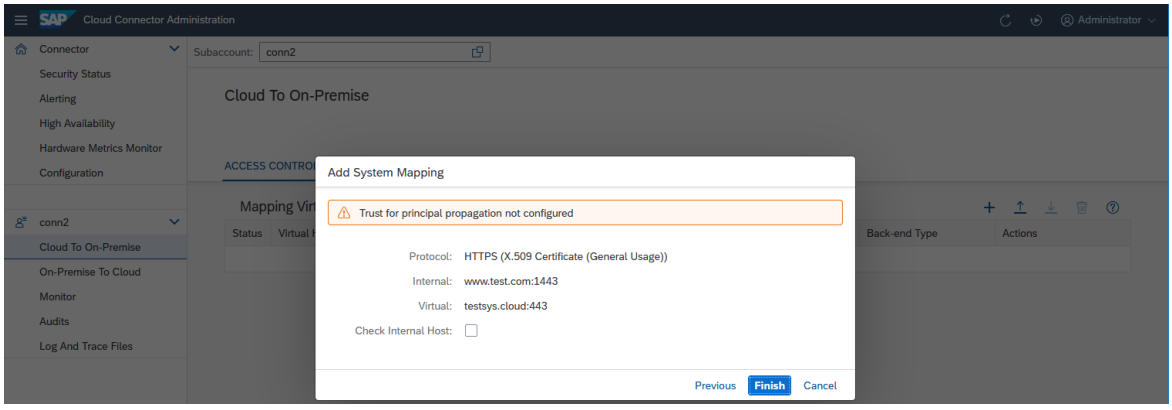
8. *Host In Request Header* lets you define, which host is used in the host header that is sent to the target server. By choosing **`Use Internal Host`**, the actual host name is used. When choosing **`Use Virtual Host`**, the virtual host is used. In the first case, the virtual host is still sent via the `X-Forwarded-Host` header.



9. You can enter an optional description at this stage. The respective description will be shown when pressing the *Info* button of the access control entry (table *Mapping Virtual to Internal System*).

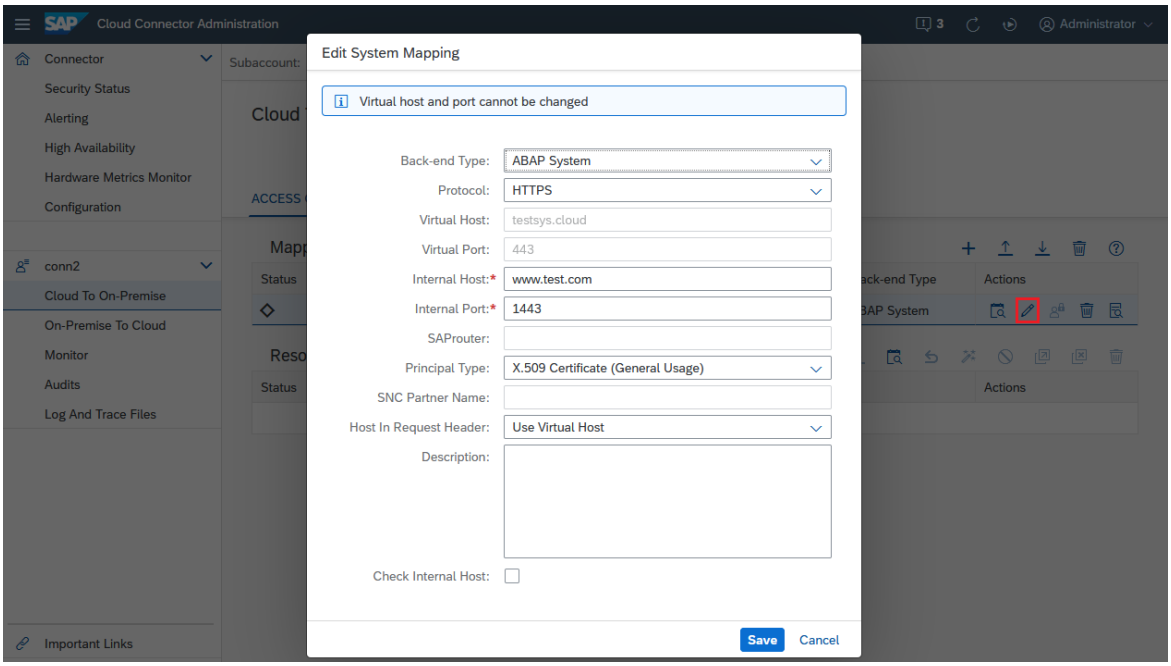


10. The summary shows information about the system to be stored and when saving the host mapping, you can trigger a ping from the Cloud Connector to the internal host, using the *Check availability of internal host* checkbox. This allows you to make sure the Cloud Connector can indeed access the internal system, and allows you to catch basic things, such as spelling mistakes or firewall problems between the Cloud Connector and the internal host. If the ping to the internal host is successful, the Cloud Connector saves the mapping without any remark. If it fails, a warning will pop up, that the host is not reachable. Details for the reason are available in the log files. You can execute such a check for all selected systems in the *Access Control* overview by pressing the button  (*Check availability...*) in column *Actions*.



11. Optional: You can later edit such a system mapping (via *Edit*) to make the Cloud Connector route the requests for `sales-system.cloud:443` to a different backend system. This can be useful if the system is currently down and there is a back-up system that can serve these requests in the meantime. However, you

cannot edit the virtual name of this system mapping. If you want to use a different fictional host name in your cloud application, you must delete the mapping and create a new one.

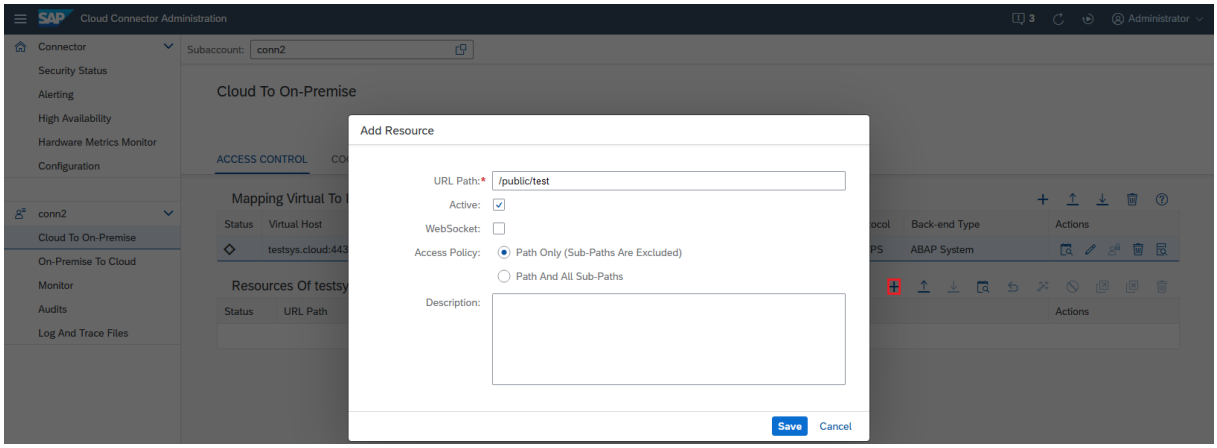


Back to Tasks [page 332]

Limit the Accessible Services for HTTP(S)

In addition to allowing access to a particular host and port, you also must specify which URL paths (*Resources*) are allowed to be invoked on that host. The Cloud Connector uses very strict allowlists for its access control. Only those URLs for which you explicitly granted access are allowed. All other HTTP(S) requests are denied by the Cloud Connector.

To define the permitted URLs for a particular backend system, choose the line corresponding to that backend system and choose *Add* in section *Resources Accessible On...* below. A dialog appears prompting you to enter the specific URL path that you want to allow to be invoked.



The Cloud Connector checks that the path part of the URL (up to but not including a possible question mark (`?`) that may denote the start of optional CGI-style query parameters) is exactly as specified in the configuration. If it is not, the request is denied. If you select option *Path and all sub-paths*, the Cloud Connector allows all requests for which the URL path (not considering any query parameters) starts with the specified string.

The *Active* checkbox lets you specify, if that resource is initially enabled or disabled. See the section below for more information on enabled and disabled resources.

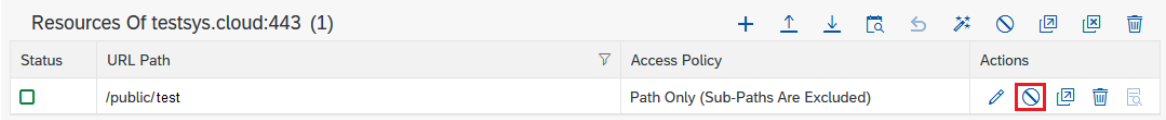
The *WebSocket Upgrade* checkbox lets you specify, whether that resource allows a protocol upgrade.

Back to Tasks [page 332]

Activate or Suspend Resources

In some cases, it is useful for testing purposes to temporarily disable certain resources without having to delete them from the configuration. This allows you to easily reprovide access to these resources at a later point of time without having to type in everything once again.

- To suspend a resource, select it and choose the *Suspend* button:
  The status icon turns red, and from now on, the Cloud Connector will deny all requests coming in for this resource.



- To activate the resource again, select it and choose the *Activate* button.
- By choosing *Allow WebSocket upgrade/Disallow WebSocket upgrade* this is possible for the protocol upgrade setting as well.
- It is also possible to mark multiple lines and then suspend or activate all of them in one go by clicking the *Activate/Suspend* icons in the top row. The same is true for the corresponding *Allow WebSocket upgrade/Disallow WebSocket* icons.

Examples:

- */production/accounting* and *Path only (sub-paths are excluded)* are selected. Only requests of the form `GET /production/accounting` or `GET /production/accounting?name1=value1&name2=value2...` are allowed. (`GET` can also be replaced by `POST`, `PUT`, `DELETE`, and so on.)
- */production/accounting* and *Path and all sub-paths* are selected. All requests of the form `GET /production/accounting-plus-some-more-stuff-here?name1=value1...` are allowed.
- */* and *Path and all sub-paths* are selected. All requests to this server are allowed.

Back to Tasks [page 332]

Related Information

Configure Domain Mappings for Cookies [page 455]
Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170]

1.4.5.2.4.2 Configure Access Control (RFC)

Specify the backend systems that can be accessed by your cloud applications using RFC.

Tasks

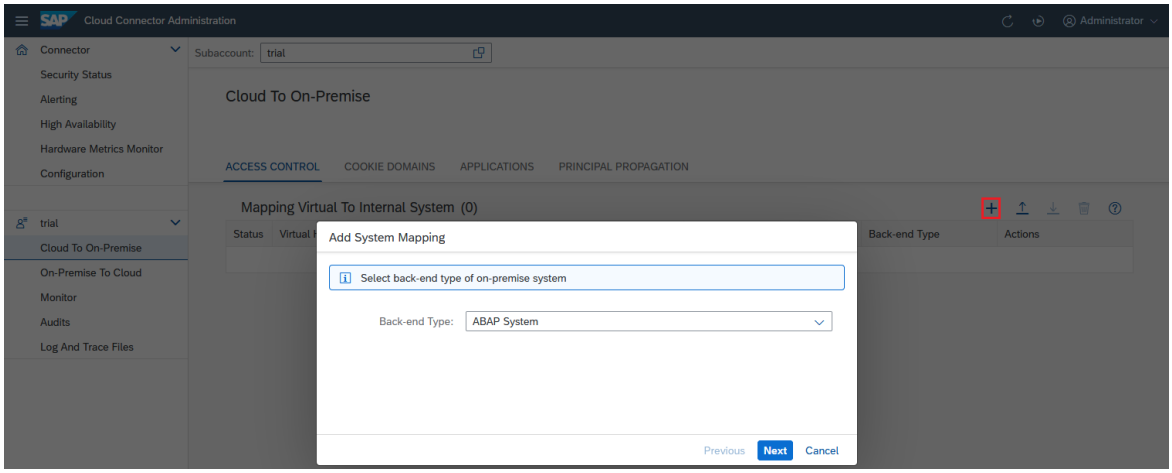
Expose Intranet Systems [page 338]

Limit the Accessible Resources for RFC [page 343]

Expose Intranet Systems

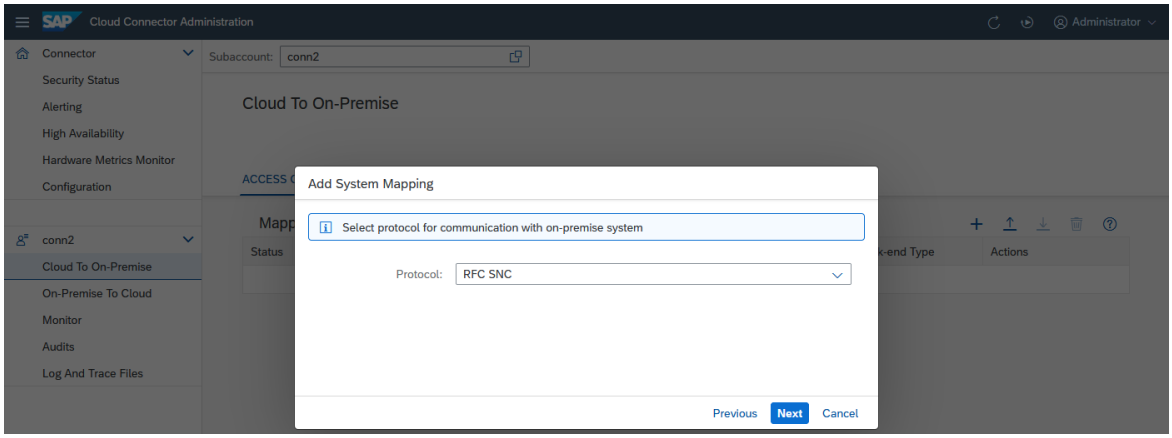
To allow your cloud applications to access a certain backend system on the intranet, insert a new entry in the Cloud Connector *Access Control* management.

1. Choose *Cloud To On-Premise* from your *Subaccount* menu and go to tab *Access Control*.
2. Choose *Add*.
3. *Backend Type*: Select the backend system type (**`ABAP System`** or **`SAP Gateway`** for RFC).



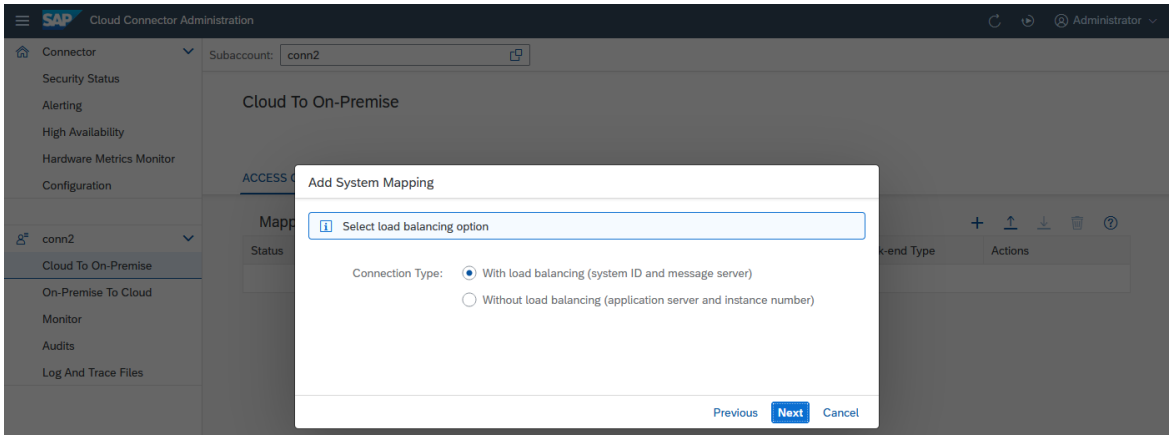
4. Choose *Next*.
5. *Protocol*: Choose **`RFC`** or **`RFC SNC`** for connecting to the backend system.

> **Note**
>
> The value **RFC SNC** is independent from your settings on the cloud side, since it only specifies the communication beween Cloud Connector and backend system. Using **RFC SNC**, you can ensure that the entire connection from the cloud application to the actual backend system (provided by the SSL tunnel) is secured, partly with SSL and partly with SNC. For more information, see Initial Configuration (RFC) [page 287].

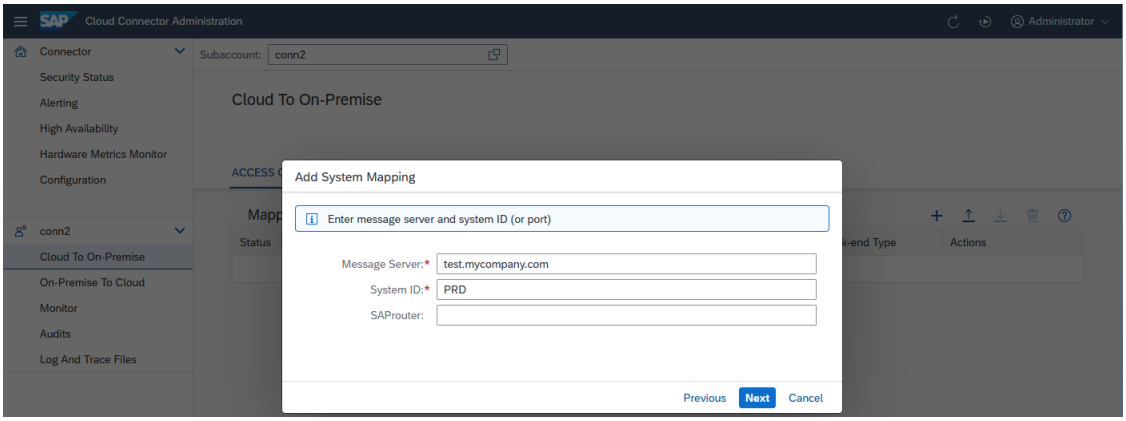


> **Note**
>
> - The back end must be properly configured to support SNC connections.
> - SNC configuration must be provided in the Cloud Connector.

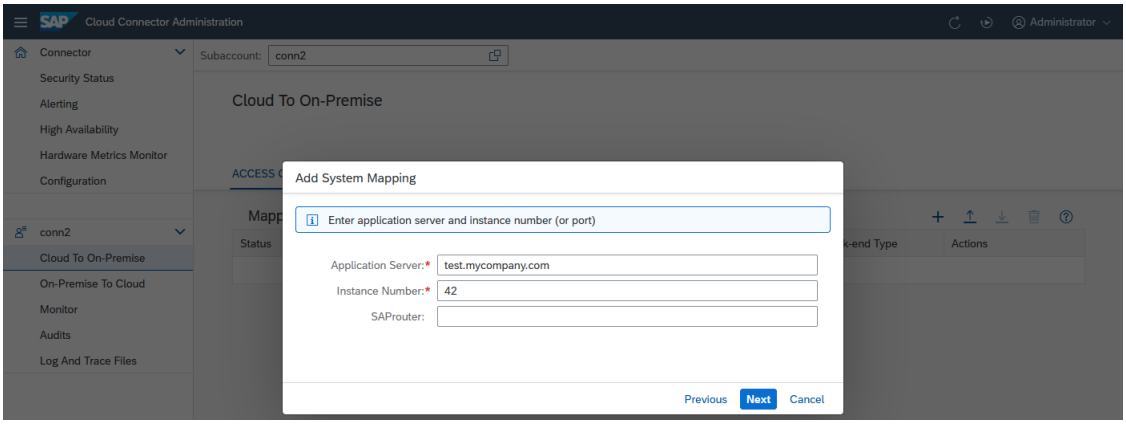
6. Choose *Next*.
7. Choose whether you want to configure a load balancing logon or connect to a specific application server.



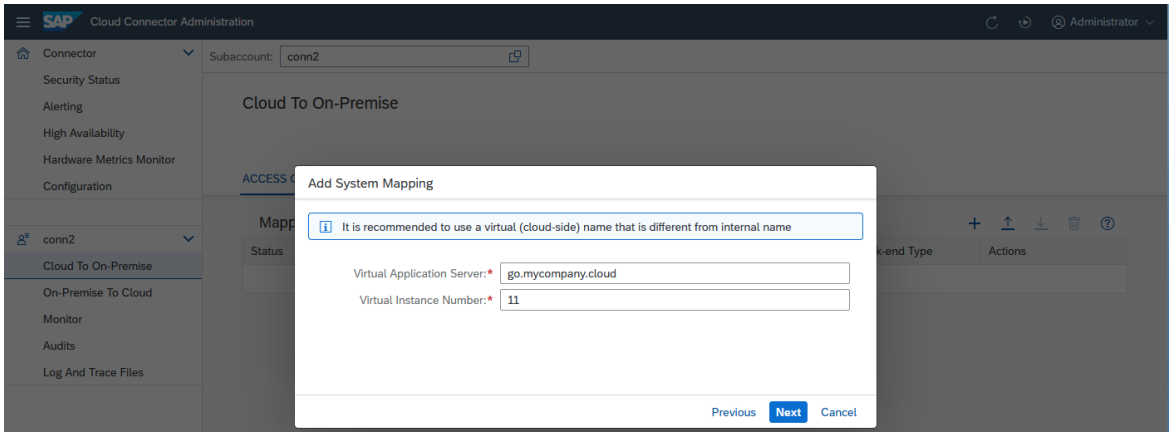
8. Specify the parameters of the backend system. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. If this is only possible using a valid SAProuter, specify the router in the respective field. The Cloud Connector will try to establish a connection to this system, so the address has to be real.
   - When using a load-balancing configuration, the *Message Server* specifies the message server of the ABAP system. The system ID is a three-char identifier that is also found in the SAP Logon configuration. Alternatively, it's possible to directly specify the message server port in the *System ID* field.



- When using direct logon, the *Application Server* specifies one application server of the ABAP system. The instance number is a two-digit number that is also found in the SAP Logon configuration. Alternatively, it's possible to directly specify the gateway port in the *Instance Number* field.



9. Optional: You can virtualize the system information in case you like to hide your internal host names from the cloud. The virtual information can be a fake name which does not need to exist. The fields will be pre-populated with the values of the configuration provided in *Message Server* and *System ID*, or *Application Server* and *Instance Number*.



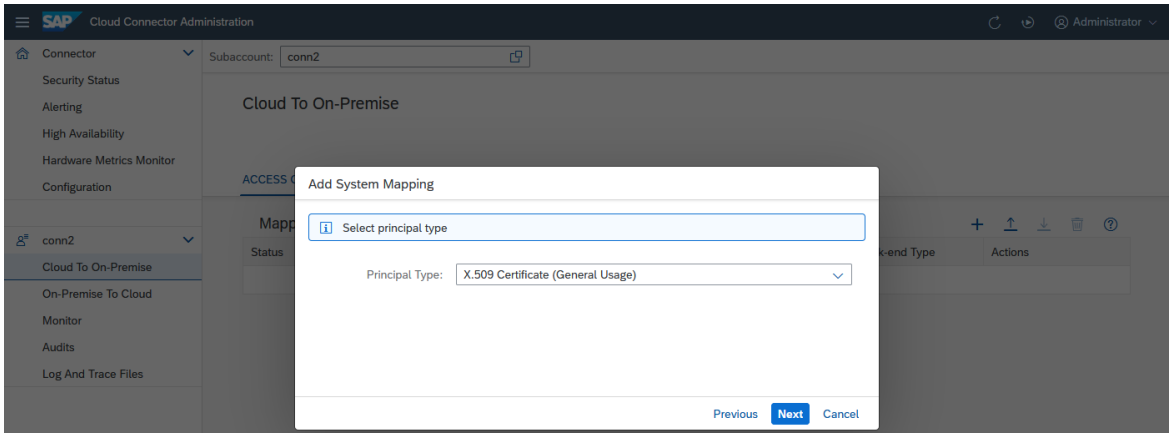
- *Virtual Message Server* - specifies the host name exactly as specified as the `jco.client.mshost` property in the RFC destination configuration in the cloud. The *Virtual System ID* allows you to distinguish between different entry points of your backend system that have different sets of access control settings. The value needs to be the same like for the `jco.client.r3name` property in the RFC destination configuration in the cloud.

- *Virtual Application Server* - it specifies the host name exactly as specified as the `jco.client.ashost` property in the RFC destination configuration in the cloud. The *Virtual Instance Number* allows you to distinguish between different entry points of your backend system that have different sets of access control settings. The value needs to be the same like for the `jco.client.sysnr` property in the RFC destination configuration in the cloud.

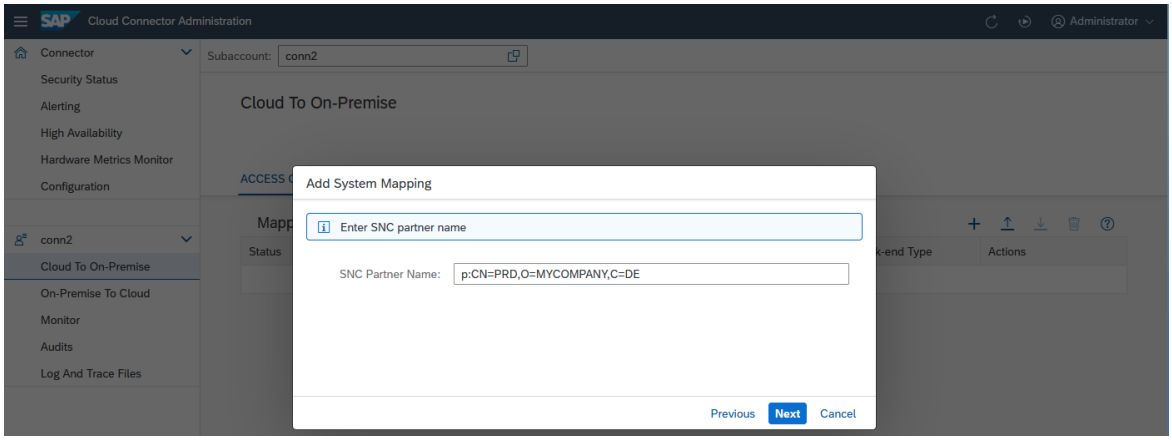
10. This step is only relevant if you have chosen **RFC SNC**. The `<Principal Type>` field defines what kind of principal is used when configuring a destination on the cloud side using this system mapping with authentication type `Principal Propagation`. No matter what you choose, make sure that the general configuration for the `<Principal Type>` has been done to make it work correctly. For destinations using different authentication types, this setting is ignored. In case you choose **None** as `<Principal Type>`, it is not possible to apply principal propagation to this system.

> **i Note**
>
> If you use an RFC connection, you cannot choose between different principal types. Only the **X.509** certificate is supported. You need an SNC-enabled backend connection to use it. For RFC, the two X.509 certificate variants *X.509 certificate (general usage)* and *X.509 certificate (strict usage)* do not differ in behavior.

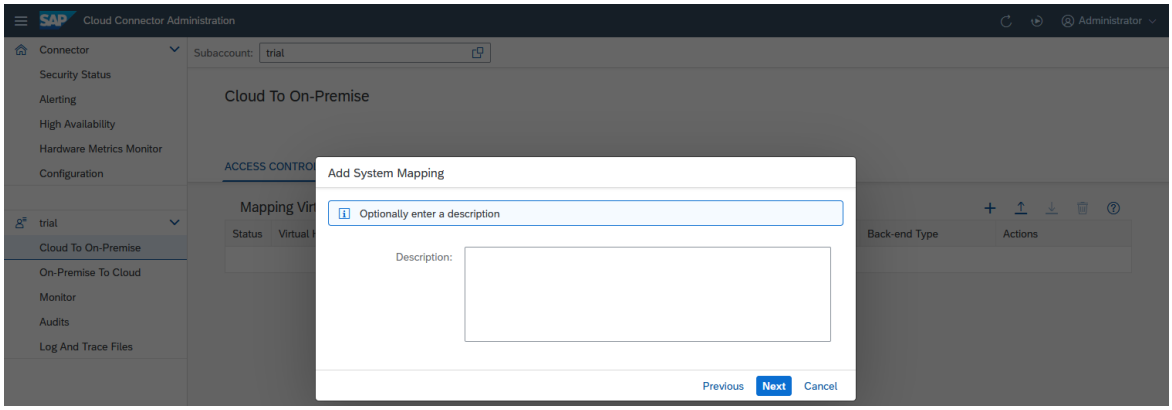
For more information on principal propagation, see Configuring Principal Propagation [page 301].



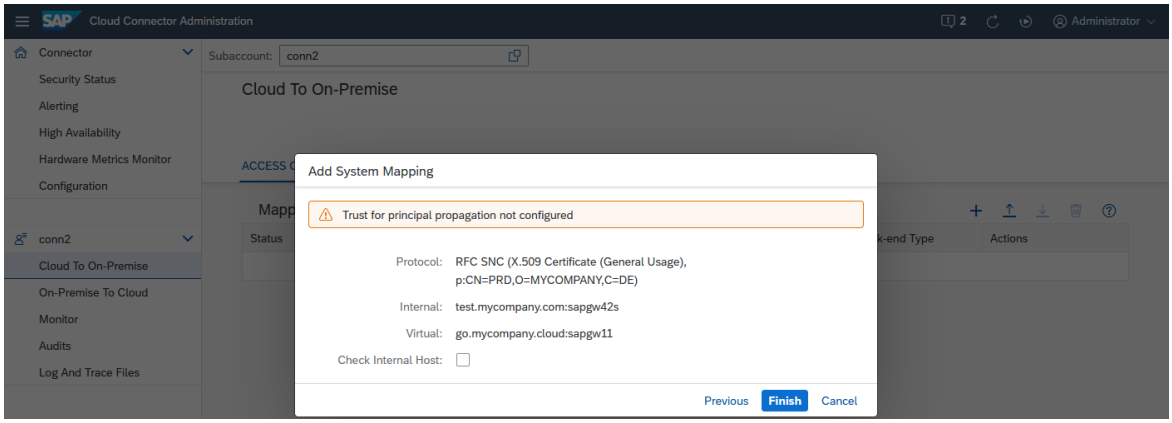
11. *SNC Partner Name*: This step will only come up if you have chosen **RFC SNC**. The SNC partner name needs to contain the correct SNC identification of the target system.



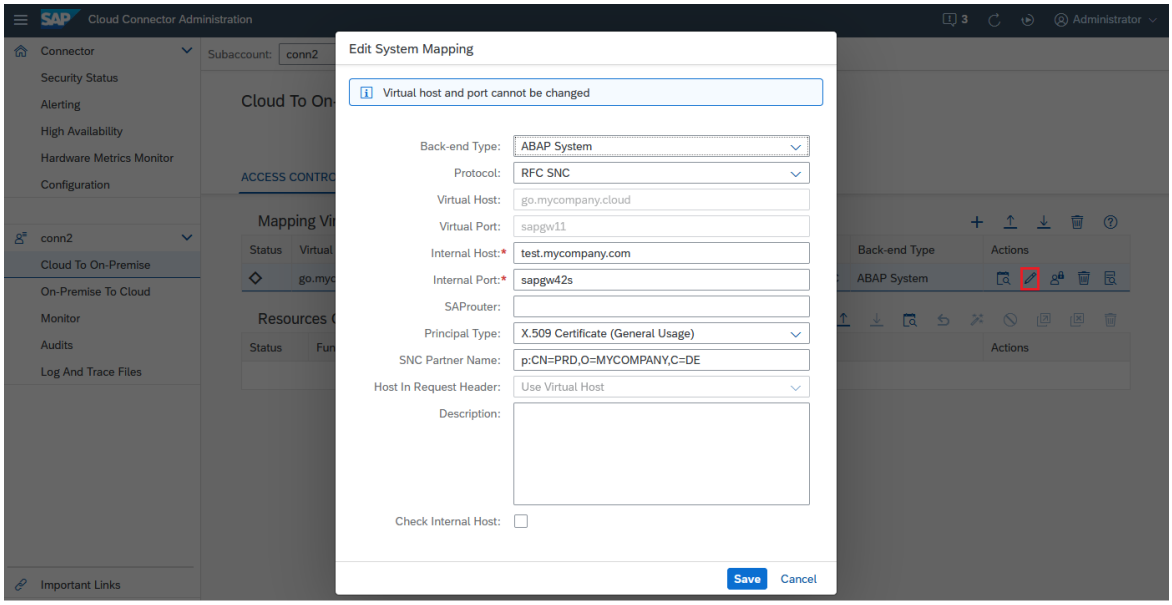
12. *Mapping Virtual to Internal System*: You can enter an optional description at this stage. The respective description will be shown as a rich tooltip when the mouse hovers over the entries of the virtual host column (table).



13. The summary shows information about the system to be stored. When saving the system mapping, you can trigger a ping from the Cloud Connector to the internal host, using the *Check availability of internal host* checkbox. This allows you to make sure the Cloud Connector can indeed access the internal system, and allows you to catch basic things, such as spelling mistakes or firewall problems between the Cloud Connector and the internal host. If the ping to the internal host is successful, the Cloud Connector saves the mapping without any remark. If it fails, a warning will pop up, that the host is not reachable. Details for the reason are available in the log files. You can execute such a check at any time later for all selected systems in the *Access Control* overview.

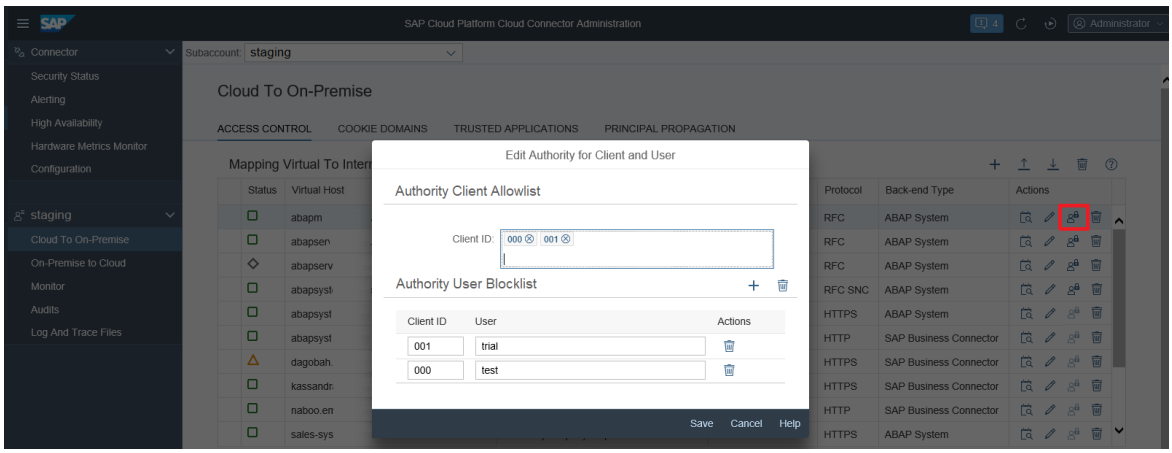


14. Optional: You can later edit a system mapping (choose *Edit*) to make the Cloud Connector route the requests for **`sales-system.cloud:sapgw42`** to a different backend system. This can be useful if the system is currently down and there is a back-up system that can serve these requests in the meantime. However, you cannot edit the virtual name of this system mapping. If you want to use a different fictional host name in your cloud application, you must delete the mapping and create a new one. Here, you can also change the *Principal Type* to **`None`** in case you don't want to allow principal propagation to a certain system.



15. *Optional*. You can later edit a system mapping to add more protection to your system when using RFC via theCloud Connector, by restricting the mapping to specified clients and users: in column *Actions*, choose the button *Maintain Authority Lists (only RFC)* to open an allowlist/blocklist dialog. In section *Authority Client Allowlist*, enter all clients of the corresponding system in the field `<Client ID>` that you want to allow to use the Cloud Connector connection. In section *Authority User Blocklist*, press the button *Add a user authority* (+) to enter all users you want to exclude from this connection. Each user must be assigned to a specified client. When you are done, press *Save*.

> **i Note**
>
> This function applies for RFC/RFC SNC only.



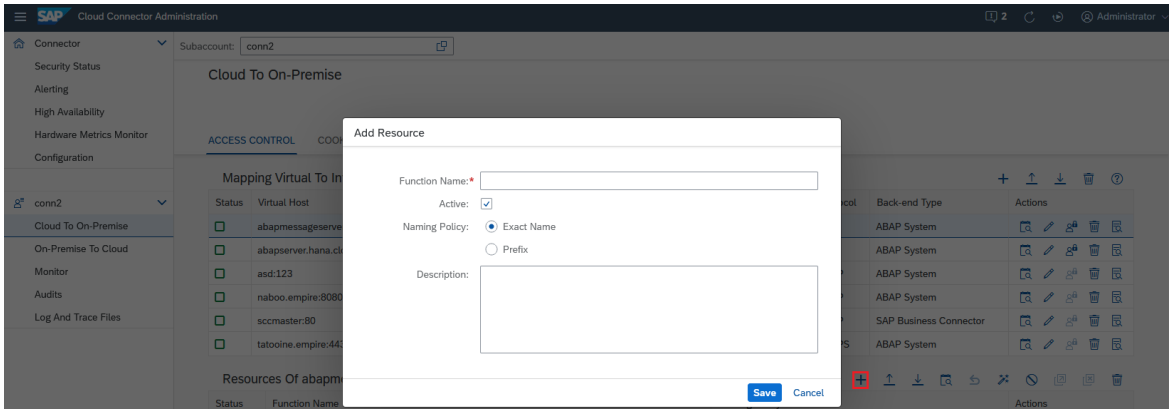
Back to Tasks [page 338]

Limit the Accessible Resources for RFC

In addition to allowing access to a particular host and port, you also must specify which function modules (*Resources*) are allowed to be invoked on that host. You can enter an optional description at this stage. The

Cloud Connector uses very strict allowlists for its access control. Besides internally used infrastructure function modules, only function modules for which you explicitly granted access are allowed.

1. To define the permitted function modules for a particular backend system, choose the row corresponding to that backend system and press *Add* in section *Resources Accessible On...* below. A dialog appears, prompting you to enter the specific function module name whose invoking you want to allow.



2. The Cloud Connector checks that the function module name of an incoming request is exactly as specified in the configuration. If it is not, the request is denied.
3. If you select the *Prefix* option, the Cloud Connector allows all incoming requests, for which the function module name begins with the specified string.
4. The *Active* checkbox allows you to specify whether that resource should be initially enabled or disabled.

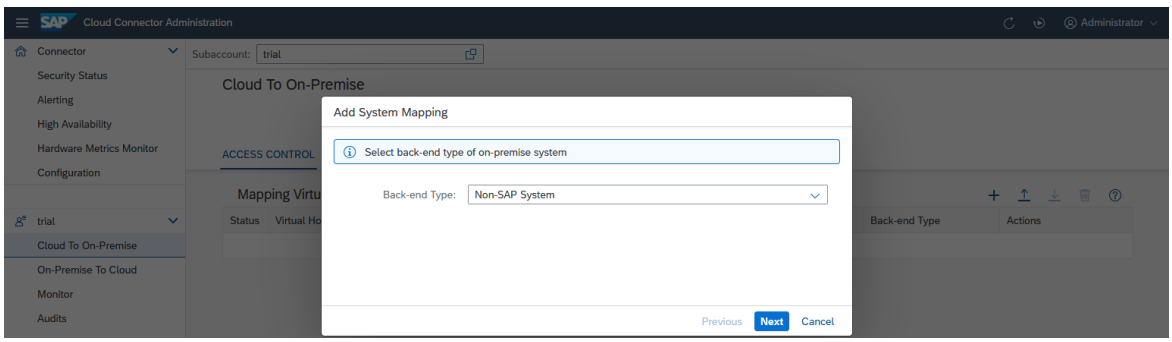
Back to Tasks [page 338]

1.4.5.2.4.3 Configure Access Control (LDAP)

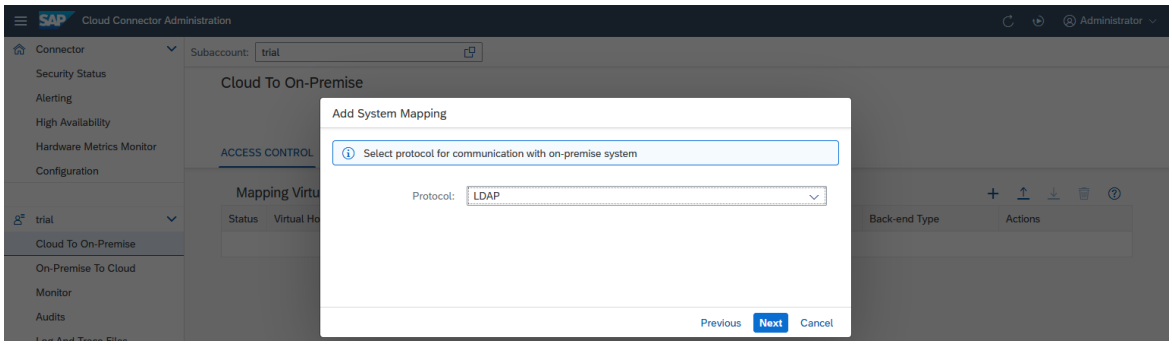
Add a specified system mapping to the Cloud Connector if you want to use an on-premise LDAP server or user authentication in your cloud application.

To allow your cloud applications to access an on-premise LDAP server, insert a new entry in the Cloud Connector access control management.

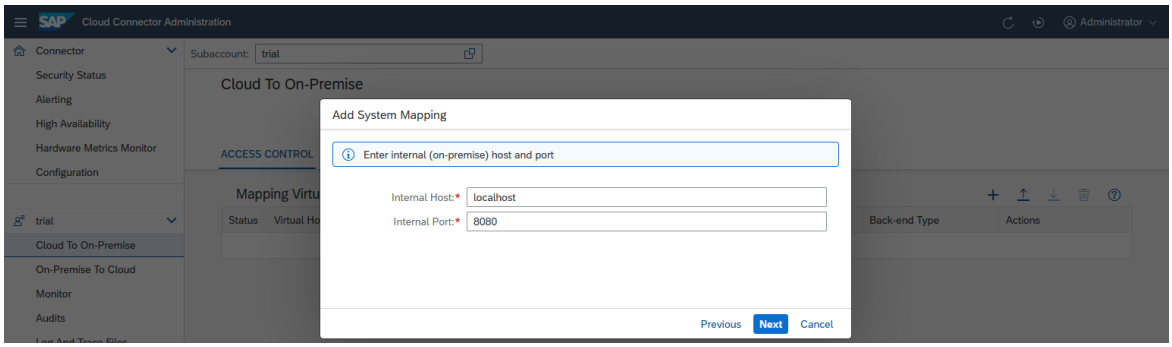
1. Choose *Cloud To On-Premise* from your *Subaccount* menu.
2. Choose *Add* (+). A wizard opens and asks for the required values.
3. *Backend Type*: Select **`Non-SAP System`** from the drop down list. When you are done, choose *Next*.



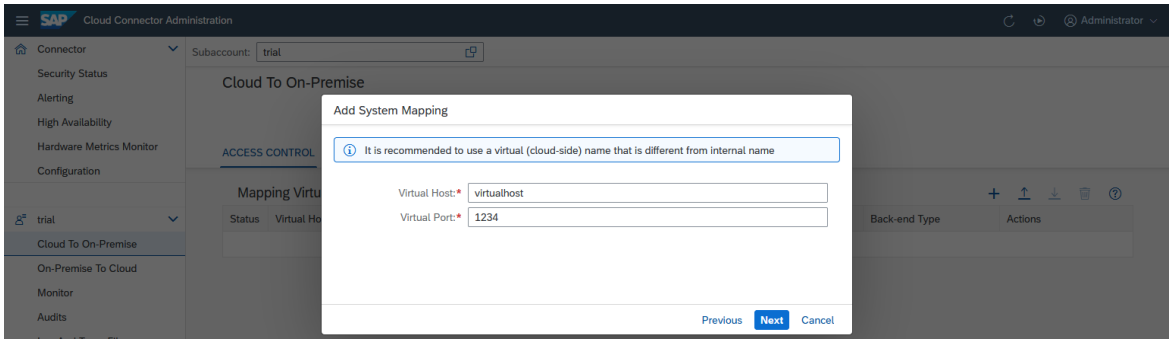
4. *Protocol*: Select **`LDAP`** or **`LDAPS`** for the connection to the backend system. When you are done, choose *Next*.



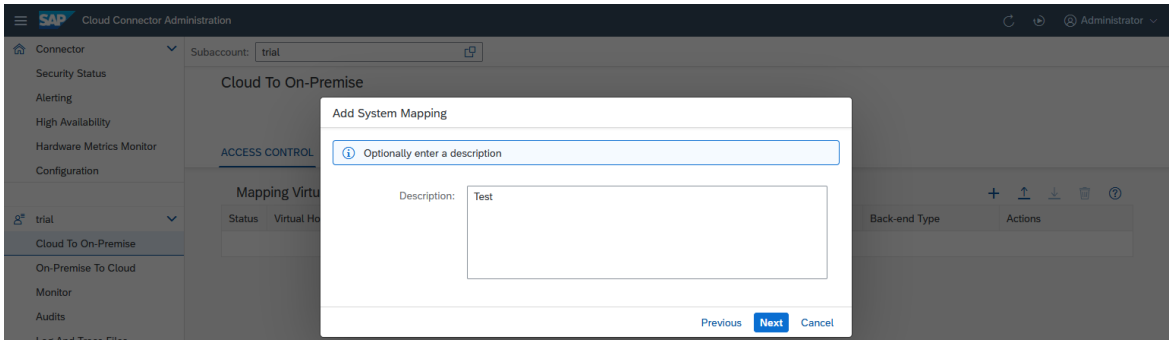
5. *Internal Host* and *Internal Port*: specify the host and port under which the target system can be reached within the intranet. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. The Cloud Connector will try to forward the request to the network address specified by the internal host and port, so this address needs to be real.



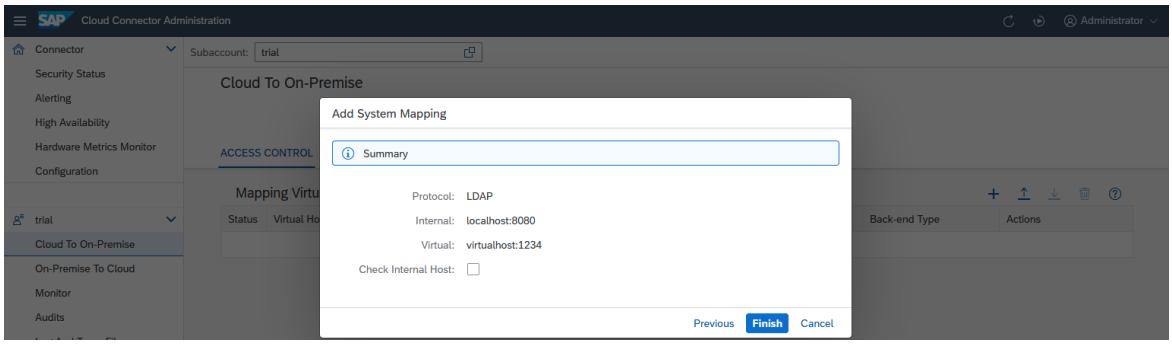
6. Enter a *Virtual Host* and *Virtual Port*. The virtual host can be a fake name and does not need to exist. The fields are pre-populated with the values of the *Internal Host* and *Internal Port*.



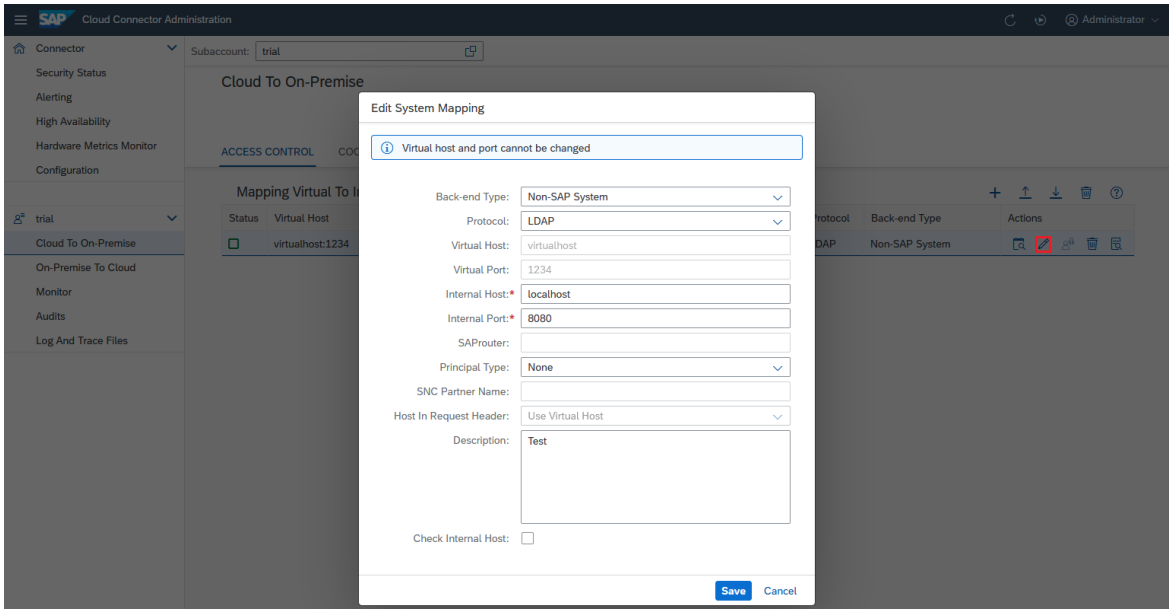
7. You can enter an optional description at this stage. The respective description will be shown as a tooltip when you press the button *Show Details* in column *Actions* of the *Mapping Virtual To Internal System* overview.



8. The summary shows information about the system to be stored. When saving the host mapping, you can trigger a ping from the Cloud Connector to the internal host, using the *Check Internal Host* check box. This allows you to make sure the Cloud Connector can indeed access the internal system. Also, you can catch basic things, such as spelling mistakes or firewall problems between the Cloud Connector and the internal host.
   If the ping to the internal host is successful, the Cloud Connector saves the mapping without any remark. If it fails, a warning is displayed in column *Check Result*, that the host is not reachable. Details for the reason are available in the log files. You can execute such a check at any time later for all selected systems in the *Mapping Virtual To Internal System* overview by pressing *Check Availability of Internal Host* in column *Actions*.



9. Optional: You can later edit the system mapping (by choosing *Edit*) to make the Cloud Connector route the requests to a different LDAP server. This can be useful if the system is currently down and there is a back-up LDAP server that can serve these requests in the meantime. However, you cannot edit the virtual name of this system mapping. If you want to use a different fictional host name in your cloud application, you have to delete the mapping and create a new one.

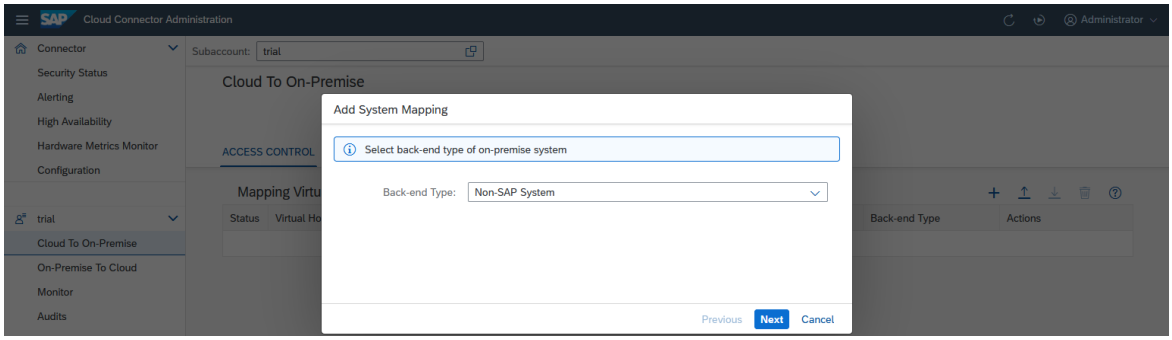


1.4.5.2.4.4 Configure Access Control (TCP)

Add a specified system mapping to the Cloud Connector if you want to use the TCP protocol for communication with a backend system.

To allow your cloud applications to access a certain backend system on the intranet via TCP, insert a new entry in the Cloud Connector access control management.

1. Choose *Cloud To On-Premise* from your *Subaccount* menu.
2. Choose *Add* (+). A wizard opens and asks for the required values.
3. *Backend Type*: Select **`Non-SAP System`** from the drop-down list. When you are done, choose *Next*.

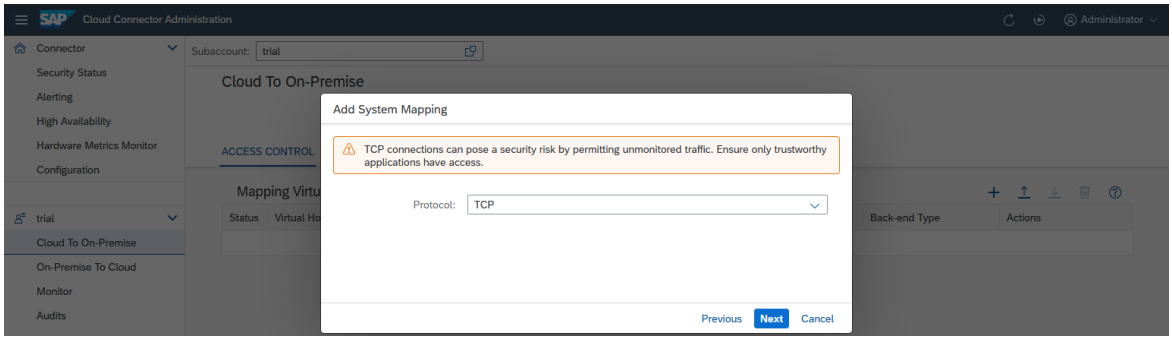


4. *Protocol*: Select **`TCP`** or **`TCP SSL`** for the connection to the backend system. When choosing TCP, you can perform an end-to-end TLS handshake from the cloud client to the backend. If the cloud-side client is using plain communication, but you still need to encrypt the hop between Cloud Connector and the backend, choose **`TCP SSL`**. When you are done, choose *Next*.

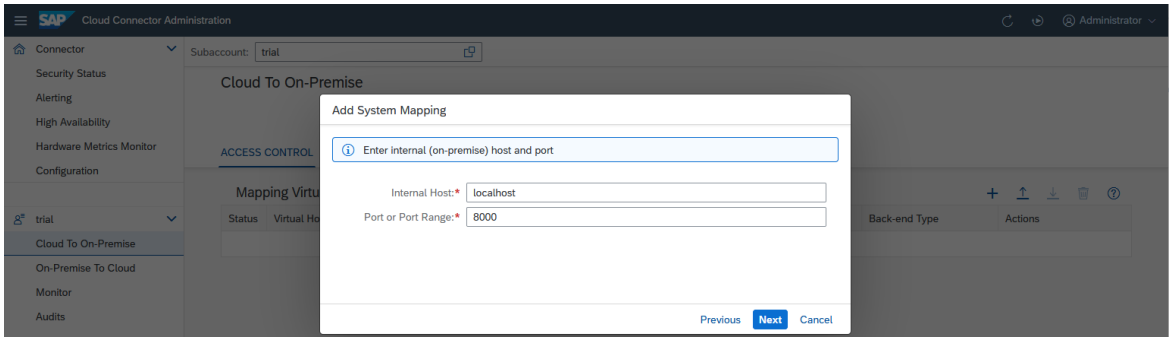
> **i Note**
>
> When selecting TCP as protocol, the following warning message is displayed: `TCP connections can pose a security risk by permitting unmonitored traffic. Ensure only`

`trustworthy applications have access`. The reason is that using plain TCP, the Cloud Connector cannot see or log any detail information about the calls. Therefore, in contrast to HTTP or RFC (both running on top of TCP), the Cloud Connector cannot check the validity of a request. To minimize this risk, make sure you

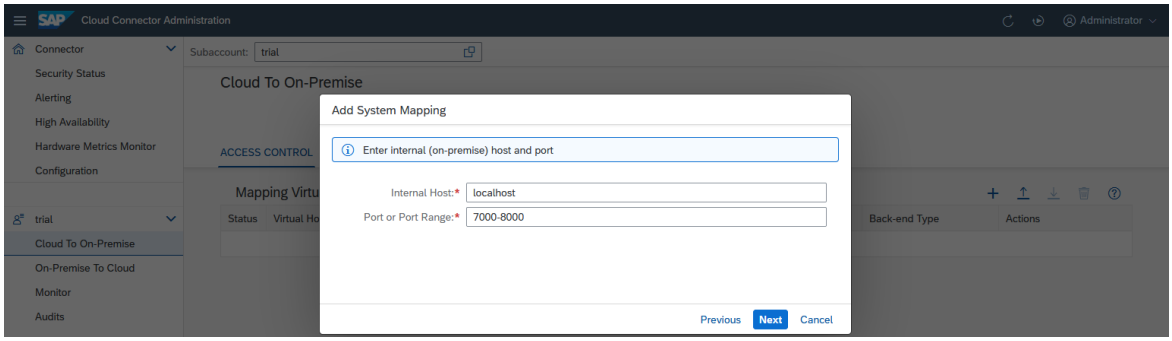
- deploy only trusted applications on SAP BTP.
- configure an application allowlist in the Cloud Connector, see Set Up Trust for Principal Propagation [page 301].
- take the recommended security measures for your SAP BTP (sub)account. See section Security in the SAP BTP documentation.



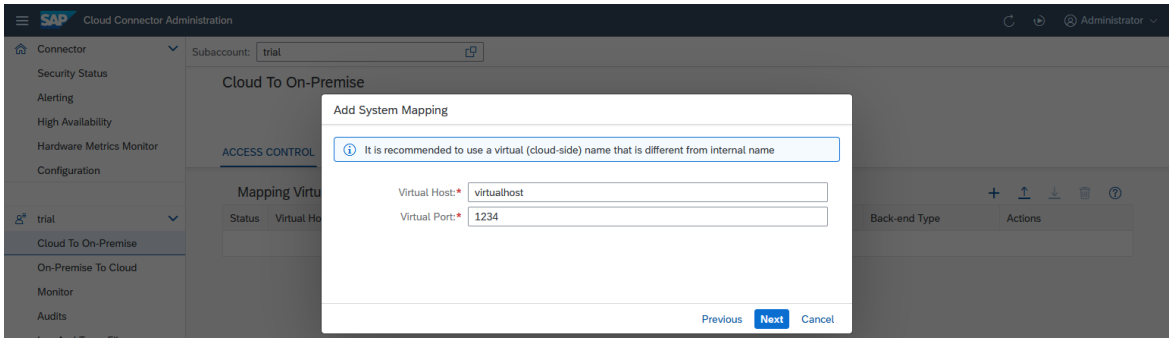
5. *Internal Host* and *Port or Port Range*: specify the host and port under which the target system can be reached within the intranet. It needs to be an existing network address that can be resolved on the intranet and has network visibility for the Cloud Connector. The Cloud Connector will try to forward the request to the network address specified by the internal host and port. That is why this address needs to be real.



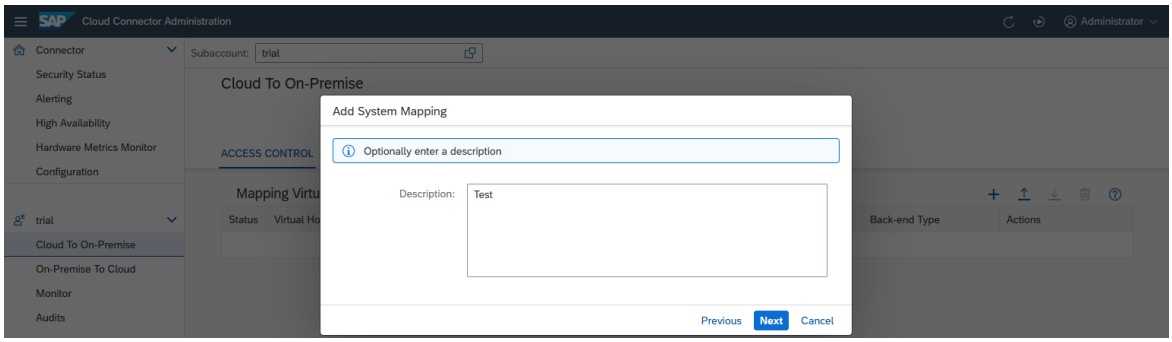
For TCP and TCP SSL, you can also specify a port range through its lower and upper limit, separated by a hyphen.



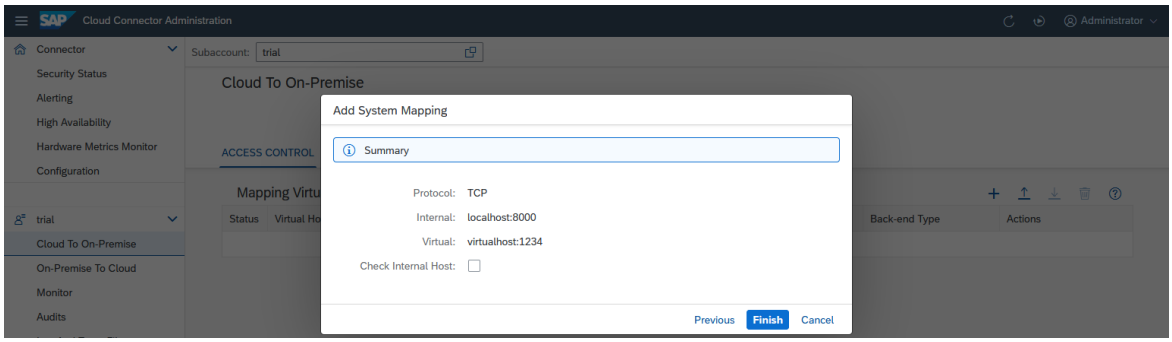
6. Enter a *Virtual Host* and *Virtual Port*. The virtual host can be a fake name and does not need to exist. The fields are prepopulated with the values of the *Internal Host* and *Port or Port Range*.



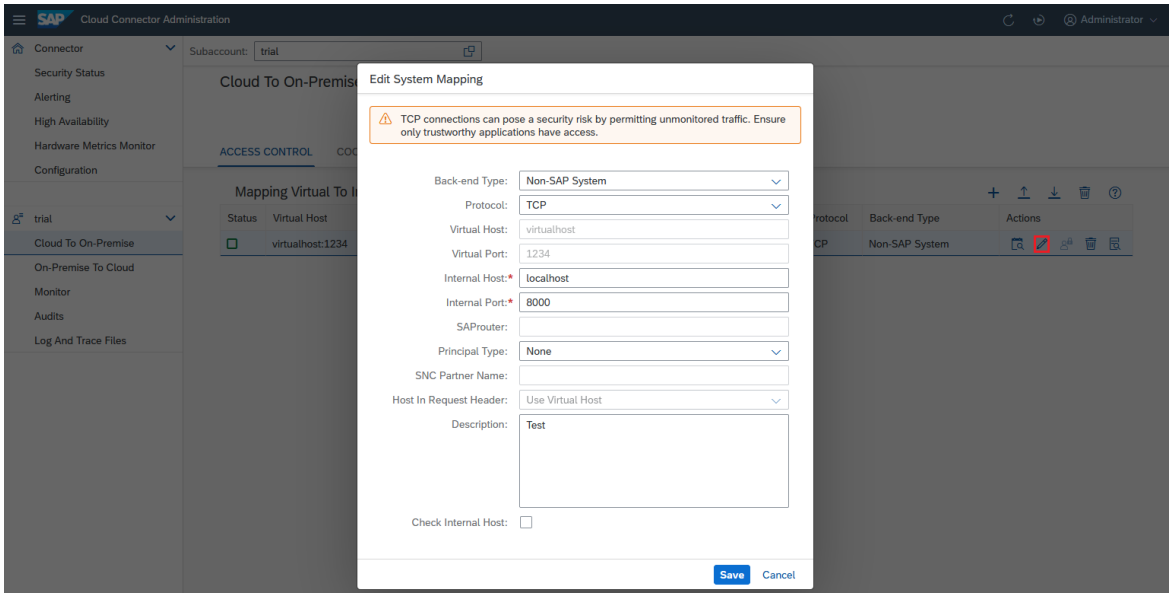
7. You can enter an optional description at this stage. The respective description will be shown as a tooltip when you press the button *Show Details* in column *Actions* of the *Mapping Virtual To Internal System* overview.



8. The summary shows information about the system to be stored. When saving the host mapping, you can trigger a ping from the Cloud Connector to the internal host, using the *Check Internal Host* checkbox. This allows you to make sure the Cloud Connector can indeed access the internal system. Also, you can catch basic things, such as spelling mistakes or firewall problems between the Cloud Connector and the internal host.
   If the ping to the internal host is successful, the Cloud Connector saves the mapping without any remark. If it fails, a warning is displayed in column *Check Result*, that the host is not reachable. Details for the reason are available in the log files. You can execute such a check at any time later for all selected systems in the *Mapping Virtual To Internal System* overview by pressing *Check Availability of Internal Host* in column *Actions*.



9. Optional: You can later edit the system mapping (by choosing *Edit*) to make the Cloud Connector route the requests to a different backend system. This can be useful if the system is currently down and there is a backup system that can serve these requests in the meantime. However, you cannot edit the virtual name nor port of this system mapping. If you want to use a different fictional host name in your cloud application, you must delete the mapping and create a new one. The same goes for port ranges. If a port range needs to be changed, you must delete the mapping and create it again with the desired port range.



1.4.5.2.4.5 Configure Accessible Resources

Configure backend systems and resources in the Cloud Connector, to make them available for a cloud application.

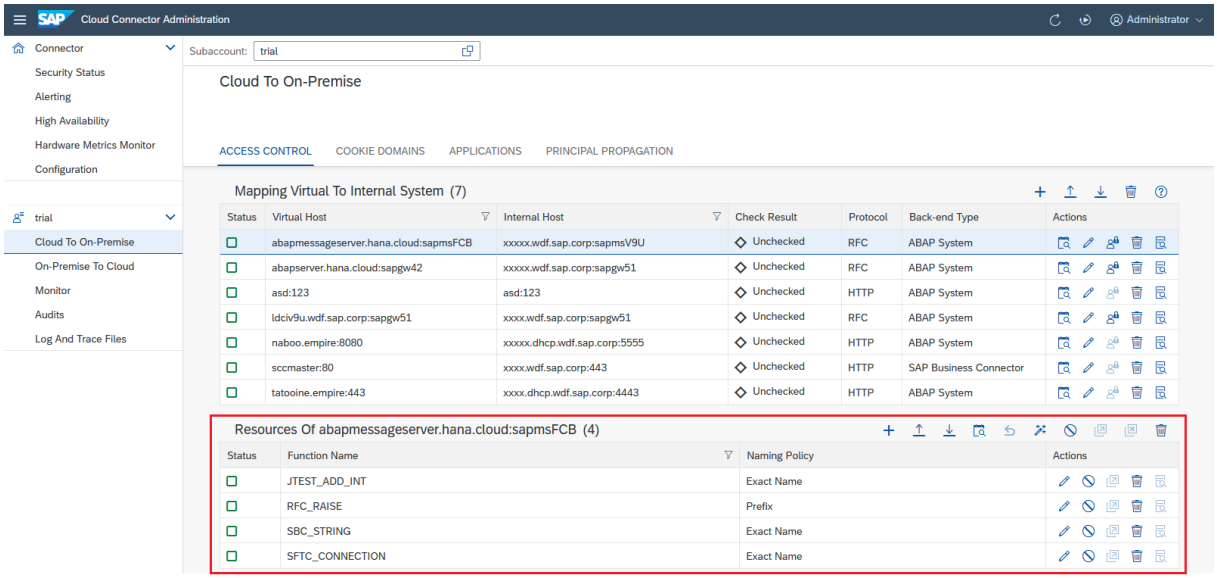
Tasks

Map Systems and Limit Access [page 350]

Use Scenarios for Resources [page 352]

Map Systems and Limit Access

Initially, after installing a new Cloud Connector, no network systems or resources are exposed to the cloud. You must configure each system and resource used by applications of the connected cloud subaccount. To do this, choose *Cloud To On Premise* from your subaccount menu and go to tab *Access Control*:

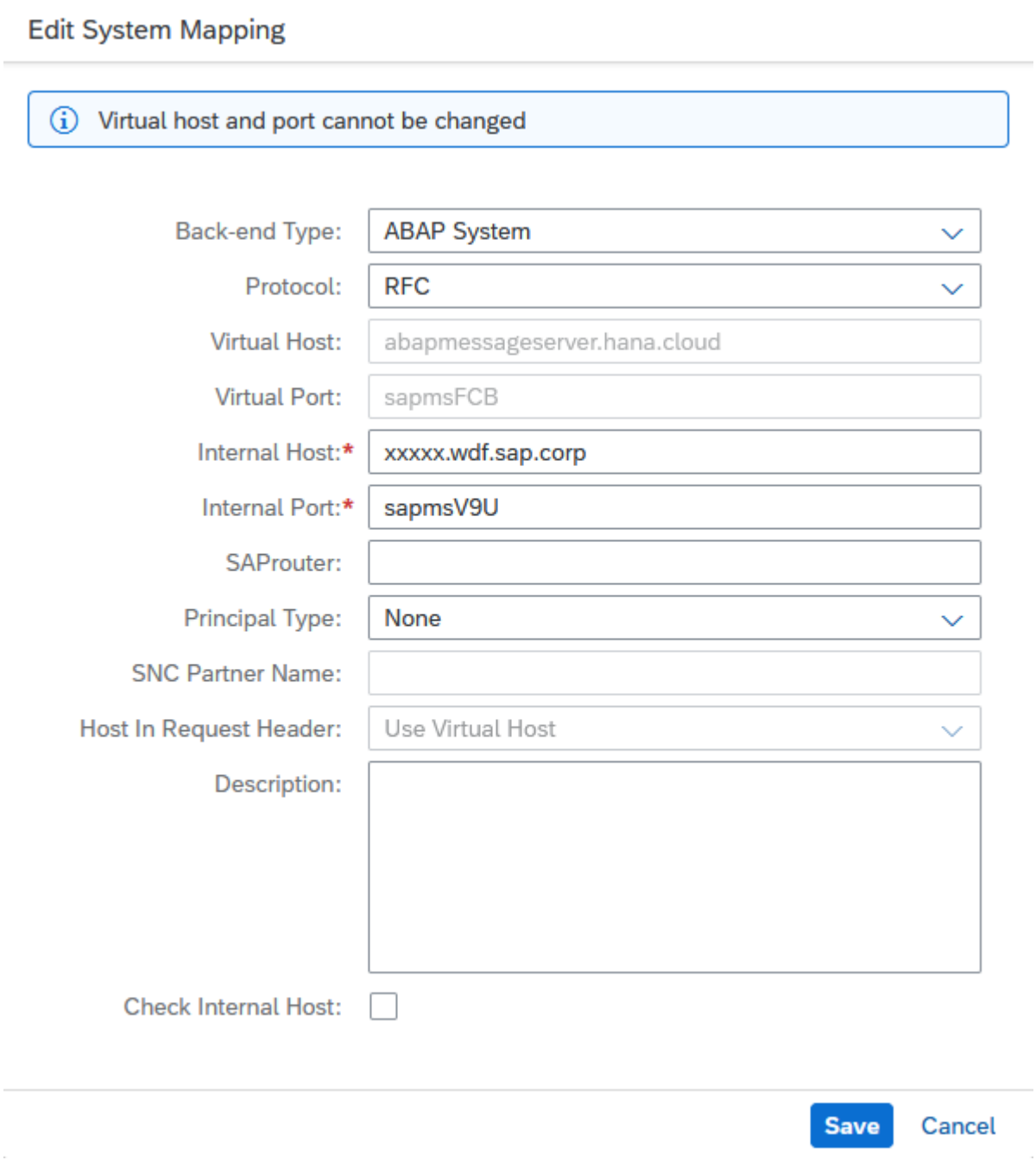


The Cloud Connector supports any type of system (SAP or non-SAP system) that can be called via one of the supported protocols. For example, a convenient way to access an ABAP system from a cloud application is via SAP NetWeaver Gateway, as it allows the consumption of ABAP content via HTTP and open standards.

- For systems using HTTP communication, see: Configure Access Control (HTTP) [page 331].
- For information on configuring RFC resources, see: Configure Access Control (RFC) [page 338].

We recommend that you limit the access to backend services and resources. Instead of configuring a system and granting access to all its resources, grant access only to the resources needed by the cloud application. For example, define access to an HTTP service by specifying the service URL root path and allowing access to all its subpaths.

When configuring an on-premise system, you can define a virtual host and port for the specified system. The virtual host name and port represent the fully qualified domain name of the related system in the cloud. We recommend that you use the virtual host name/port mapping to prevent leaking information about a system's physical machine name and port to the cloud.



Back to Tasks [page 350]

Use Scenarios for Resources

As of version 2.12, the Cloud Connector lets you define a set of resources as a scenario that you can export, and import into another Cloud Connector.

Scenarios are useful if you provide an application to many consumers, which invokes a large number of resources in an on-premise system. In this case, you must expose a system on your Cloud Connector that covers all required resources, which increases the risk of incorrect configuration.

Define and Export a Scenario [page 353]

Import a Scenario [page 353]

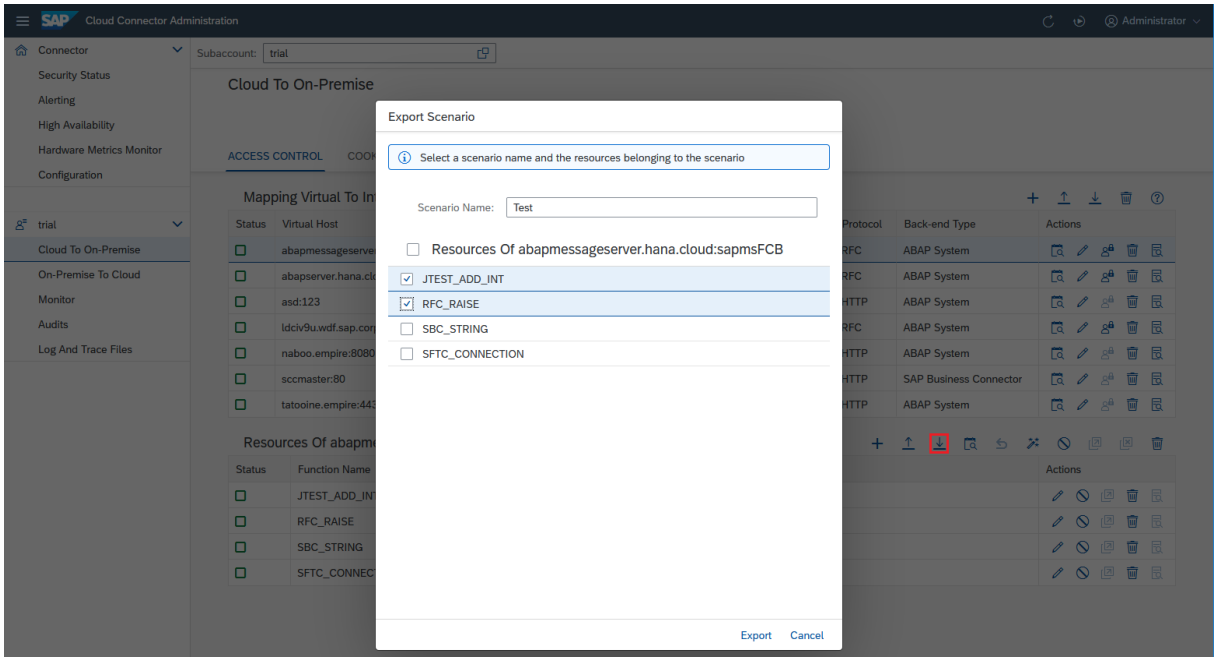
Remove a Scenario [page 354]

Define and Export a Scenario

If you, as application owner, have implemented and tested a scenario, and configured a Cloud Connector accordingly, you can define the scenario as follows:

1. Choose the *Export Scenario* button.
2. In the dialog, select the resources that belong to the scenario.
3. In the field `<Scenario Name>`, choose a descriptive name, under which the set of resources can be identified in the consuming Cloud Connector.
4. Choose *Export* to store the scenario.

> **i Note**
>
> For applications provided by SAP, default scenario definitions may be available. To verify this, check the corresponding application documentation.



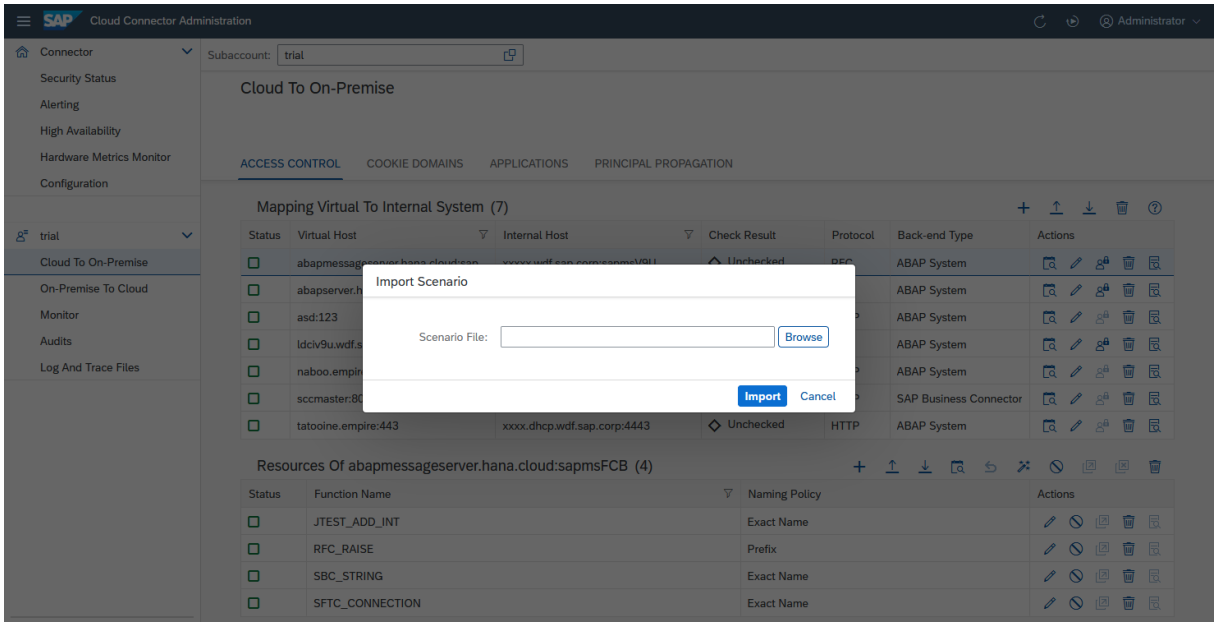
Back to Use Scenarios for Resources [page 352]

Back to Tasks [page 350]

Import a Scenario

As an administrator taking care for a scenario configuration in some other Cloud Connector, perform the following steps:

1. Choose the *Import Scenario* button to add all required resources to the desired access control entry.
2. In the dialog, navigate to the folder of the archive that contains the scenario definition.
3. Choose *Import*. The resources of the scenario are merged with the existing set of resources which are already available in the access control entry.



All resources belonging to a scenario get an additional scenario icon in their status. When hovering over it, the assigned scenario(s) of this resource are listed.

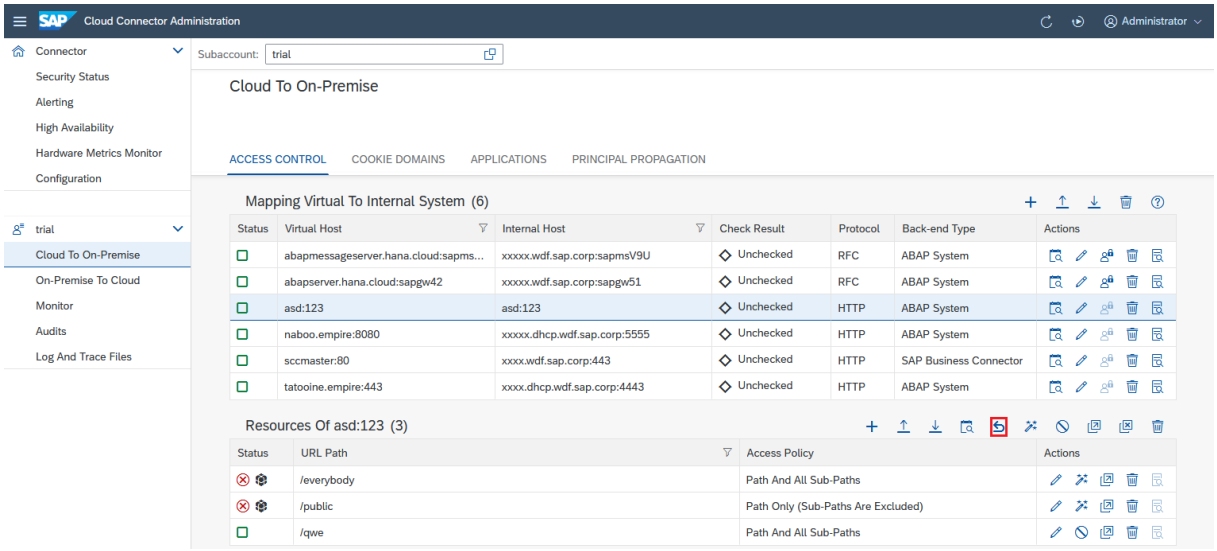
Back to Use Scenarios for Resources [page 352]

Back to Tasks [page 350]

Remove a Scenario

To remove a scenario:

1. Choose the *Remove Scenario* button.
2. In the dialog, choose the scenario(s) you want to remove.
3. Choose *Remove* to remove all resources associated with the chosen entry from the access control entry. Resources that existed before, or are assigned to another scenario as well, are not removed.



Back to Use Scenarios for Resources [page 352]

Back to Tasks [page 350]

1.4.5.2.5 Configuration REST APIs

You can use a set of APIs to perform the basic setup of the Cloud Connector.

Context

As of version 2.11, the Cloud Connector provides several REST APIs that let you configure a newly installed Cloud Connector. The configuration options correspond to the following steps:

- Initial Configuration [page 278]
- Managing Subaccounts [page 289]
- Configure Access Control [page 330]
- Master and Shadow Administration [page 473]

All configuration APIs start with the following string: `/api/v1/configuration`.

> **i Note**
>
> Use the same host and port for the REST APIs that you use to access the Cloud Connector.

Prerequisites

- After installing the Cloud Connector, you have changed the initial password.
- You have specified the high availability role of the Cloud Connector (master or shadow).
- You have configured the proxy on the master instance if required for your network.

Request and Response Format

Requests and responses are coded in JSon format. The following example shows the request payload `{description:<value>}` coded in JSon:

> Sample Code
>
> ```
> {"description":"very helpful description"}
> ```

Values that represent a date are given as a UTC long number, which is the number of milliseconds since 1 January 1970 00:00:00 UT (GMT+00).

In case of errors, the HTTP status code is 4xx. Error details are supplied in the response body in JSON format:

```
{type: <type>, message: <message>}
```

`type` can have one of these values:

`INVALID_REQUEST`, `ILLEGAL_STATE`, `NOT_FOUND`, `INVALID_CONFIGURATION`, `RUNTIME_FAILURE`, `SHADOW_UPDATE`, `FORBIDDEN_REQUEST`

Security and Sessions

The Cloud Connector supports basic authentication and form-based authentication. Once authenticated, the client can keep the session and execute subsequent requests in this session. A session avoids the overhead caused by authentication.

You can get the session ID from the response header field `<Set-cookie>` (as `JSESSIONID=<session ID>`), and send it in the request header `Cookie: JSESSIONID=<session Id>`.

The Cloud Connector uses CSRF tokens to prevent CSRF (cross-site request forgery) attacks. Upon first request, a CSRF token is generated and sent back in the response header in field `<X-CSRF-Token>`. The client application must keep this token and send it in all subsequent requests as header field `<X-CSRF-Token>`, together with the session cookie as described above.

> Note
>
> If the request header field `<Connection>` has the value `close` (as opposed to `keep-alive`), no CSRF token is generated. If you want to make stateful, session-based REST calls, use `Connection: keep-`

`alive`, extract the CSRF token from the first response header, and include it in all request headers. Otherwise, use `Connection: close` and always submit user and password through basic authentication.

An inactive session causes a timeout at some point, and will consequently be removed. A request using an expired session receives a login page (`Content-type: text/html`). The status code of the response is 200 in this case. The only way to detect an expired session is to check the content type and status code. Content type *text/html* in a connection with status code *200* indicates an expired session.

For security reasons, a session should be closed or invalidated once it is not needed anymore. You can achieve this by including `Connection: close` in the header of the final call of the relevant session. As a result, the Cloud Connector invalidates the session. Subsequent attempts to send a request in the context of that session will respond with a login page as described above.

User Roles

As of Cloud Connector 2.12, the REST API supports different user roles. Depending on the role, an API grants or denies access. In default configuration, the Cloud Connector uses local user storage and supports the single user `Administrator` (administrator role). Using LDAP user storage, you can use various users (see also Configure Named Cloud Connector Users [page 465]):

Role	Technical Role Name	Authorization
Administrator	`admin` or `sccadmin`	All CRUD operations.
Support	`sccsupport`	Edit log and trace configuration.
Display	`sccdisplay`	Read configuration.
Monitoring	`sccmonitoring`	Read monitoring information.

Return Codes

Successful requests return the code 200, or, if there is no content, 204. *POST* actions that create new entities return 201, with the location link in the header.

The following general errors can be returned by each API:

400 – invalid request. For example, if parameters are invalid or the API is not supported anymore, an unexpected state occurs and in case of other non-critical errors.

401 – authorization required.

403 – the current Cloud Connector instance does not allow changes. For example, the instance has been assigned to the shadow role and therefore does not allow configuration changes, or the user role does not have required permission.

405 – an entity does not support the requested operation.

409 – current state of the Cloud Connector does not allow changes. For example, the password has to be changed first.

415 – unexpected mime type.

500 – operation failed.

The status line provides details explaining the reason.

Most APIs may also return specific error details, depending on the situation. Such errors are mentioned in the corresponding API description.

> **i Note**
>
> The error texts depend on the request locale.

Hypertext Application Language

Entities returned by the APIs contain links as suggested by the current draft *JSON Hypertext Application Language* (see https://tools.ietf.org/html/draft-kelly-json-hal-08).

Available APIs

API Name	Use
Common Properties [page 360]	• Read common properties • Edit common properties
High Availability Settings [page 361]	• Get high availaility settings • Set high availaility settings
Proxy Settings [page 372]	• Get proxy settings • Set proxy settings • Remove proxy settings
Authentication and UI Settings [page 375]	• Read authentication settings • Edit authentication settings • Edit LDAP authentication settings

API Name	Use
Certificate Management for Backend Communication [page 383]	• Get description for CA certificate for principal propagation • Get binary content of CA certificate for principal propagation • Create a self-signed CA certificate for principal propagation • Create a certificate signing request for CA certificate for principal propagation • Upload a signed certificate chain as CA certificate for principal propagation • Upload a PKCS#12 certificate as CA certificate for principal propagation • Delete CA certificate for principal propagation • Get description for system certificate • Get binary content of system certificate • Create a self-signed system certificate • Create a certificate signing request for a system certificate • Upload a signed certificate chain as system certificate • Upload a PKCS#12 certificate as system certificate • Delete system certificate
Solution Management Configuration [page 396]	• Get solution management configuration • Set solution management configuration and turn on reporting • Turn off reporting • Download current LMDB XML report
Backup [page 398]	• Create backup configuration • Restore backup configuration
Subaccount [page 400]	• Get list of subaccounts • Create subaccount • Delete subaccount • Edit subaccount • Connect or disconnect subaccount • Extend validity of subaccount • Create or change recovery subaccount • Delete recovery subaccount • Extend validity of recovery subaccount • Get subaccount configuration
System Mappings [page 409]	• Get system mappings • Create system mapping • Delete system mapping • Delete all system mappings • Edit system mapping

API Name	Use
System Mapping Resources [page 414]	• Get system mapping resources • Get system mapping resource • Create system mapping resource • Edit system mapping resource • Delete system mapping resource • Delete all system mapping resources
Domain Mappings [page 418]	• List domain mappings • Create domain mappings • Delete domain mappings • Edit domain mappings
Subaccount Service Channels [page 421]	• Get service channels • Create service channel • Delete service channel • Edit service channel • Enable or disable service channel

Related Information

Examples [page 424]

1.4.5.2.5.1 Common Properties

Read and edit the Cloud Connector's common properties via API.

Get Common Properties

URI	`/api/v1/configuration/connector`
Method	*GET*
Request	
Response	`{ha, description}`
Errors	

Roles	Administrator, Display, Support

- **Response Properties**:
  - `ha`: Cloud Connector instance assigned to a high-availability role. Its value is either the string *master* or *shadow*.
  - `description`: Description of the Cloud Connector.

Set Description (Master Only)

URI	`/api/v1/configuration/connector`
Method	*PUT*
Request	`{description}`
Response	`{ha, description}`
Errors	`INVALID_REQUEST`
Roles	Administrator

- **Request Properties**:
  `description`: A string; use an empty string to remove the description.
- **Response Properties**:
  - `ha`: Cloud Connector instance assigned to a high-availability role. Its value is either the string *master* or *shadow*.
  - `description`: Description of the Cloud Connector.
- **Errors**:
  `INVALID_REQUEST` ((400): The value of `description` is not a JSON string.

  > **i** Note
  >
  > `null` is not a JSON string.

1.4.5.2.5.2 High Availability Settings

Read and edit the high availability settings of a Cloud Connector instance via API.

When installing a Cloud Connector instance, you usually define its high availability role (master or shadow instance) during initial configuration, see Change your Password and Choose Installation Type [page 280].

If the high availability role was not defined before, you can set the master or shadow role via this API.

If a shadow instance is connected to the master, this API also lets you switch the roles: the master instance requests the shadow instance to take over the master role, and then takes the shadow role itself.

> **i** Note
>
> Editing the high availability settings is only allowed on the master instance, and supports only **`shadow`** as input.

Get High Availability Settings

URI	`/api/v1/configuration/connector/haRole`
Method	*GET*
Request	
Response	"<HA role>"
Errors	
Roles	Administrator, Display, Support

Example

```
curl -i -k -u Administrator:<password> https://localhost:8443/api/v1/
configuration/connector/haRole
```

Set High Availability Settings

Use this API if you want to set the role of a fresh installation (no role assigned yet).

As of version 2.12.0, this API also allows to switch the roles if a shadow instance is connected to the master. In this case, the API is only allowed on the master instance and supports only the value **`shadow`** as input. The master instance requests the shadow instance to take over the master role and then assumes the shadow role itself.

URI	`/api/v1/configuration/connector/haRole`
Method	*POST*
Request	"master" or "shadow"
Response	
Errors	ILLEGAL_STATE, INVALID_REQUEST
Roles	Administrator

Errors:

- `INVALID_REQUEST` (400): If a high-availability role other than "master" or "shadow" is supplied.
- `ILLEGAL_STATE` (409): If changing the high-availability role is not possible; changing the role is only possible if no role has been assigned, or if the current role is master and a shadow system is connected.

Example

```
curl -i -k -H 'Content-Type: application/json' -d '"shadow"'  -u
Administrator:<password> -X POST https://localhost:8443/api/v1/configuration/
connector/haRole
```

Related Information

Master Instance Configuration [page 363]
Shadow Instance Configuration [page 367]

1.4.5.2.5.2.1 Master Instance Configuration

Read and edit the high availability settings for a Cloud Connector master instance via API.

> **i** Note
>
> The APIs below are available as of Cloud Connector version 2.13.0.

> **!** Restriction
>
> These APIs are only permitted on a Cloud Connector master instance. The shadow instance rejects the requests with error code *400 – Invalid Request*.

Get Configuration

URI	`/api/v1/configuration/connector/ha/master/config`
Method	*GET*
Request	
Response	`{haEnabled, allowedShadowHost}`
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `haEnabled`: a Boolean value that indicates whether or not a shadow system is allowed to connect
- `allowedShadowHost`: the name of the shadow host (a string) that is allowed to connect; an empty string signifies that any host is allowed to connect as shadow.

Example

```
curl -i -k -u <user>:<password> -X GET https://host>:<port>/api/v1/configuration/connector/ha/master/config
```

Set Configuration

URI	`/api/v1/configuration/connector/ha/master/config`
Method	*PUT*
Request	`{haEnabled, allowedShadowHost}`
Response	204 on success

Errors	INVALID_REQUEST
Roles	Administrator

Response Properties:

- `haEnabled`: Boolean value that indicates whether or not a shadow system is allowed to connect.
- `allowedShadowHost`: Name of the shadow host (a string) that is allowed to connect. An empty string means that any host is allowed to connect as shadow.

Errors:

- `INVALID_REQUEST` (400): if the name of the shadow host is not a valid host name

Example

```
curl -i -k -u <user>:<password> -X PUT -H 'Content-Type: application/json' -d
'{"haEnabled":true}' https://<host>:<port>/api/v1/configuration/connector/ha/
master/config
```

Get State

URI	`/api/v1/configuration/connector/ha/master/state`
Method	*GET*
Request	
Response	`{state, shadowHost}`
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `state`: One of the following strings: ALONE, BINDING, CONNECTED or BROKEN.
- `shadowHost`: Connected shadow host (a string)

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/
configuration/connector/ha/master/state
```

Set State

URI	`/api/v1/configuration/connector/ha/master/state`
Method	*POST*
Request	
Response	`{op}`
Errors	ILLEGAL_STATE, INVALID_REQUEST
Roles	Administrator

Request Properties:

The value of property `op` is one of the following strings:

- `SWITCH`: Switch roles with shadow
- `FORCE_SWITCH`: Take over the shadow role, even if shadow instance does not respond.

Errors:

- `INVALID_REQUEST` (400): Value of property `op` is neither SWITCH nor FORCE_SWITCH.
- `ILLEGAL_STATE` (409): Master system is in a state that does not permit the requested operation.

Example

```
curl -i -k -u <user>:<password> -X POST -H 'Content-Type: application/json' -d
'{"op":"SWITCH"}' https://<host>:<port>/api/v1/configuration/connector/ha/master/
state
```

Reset

A successful call to this API restores default values for all settings related to high availability on the master side.

> ⚠ Caution
>
> Do not perform this call if the shadow is connected to a master.

URI	`/api/v1/configuration/connector/ha/master/state`
Method	*DELETE*
Request	
Response	204 on success
Errors	ILLEGAL_STATE
Roles	Administrator

Errors:

- `ILLEGAL_STATE` (409): A shadow instance is connected to the master.

Example

```
curl -i -k -u <user>:<password> -X DELETE https://<host>:<port>/api/v1/
configuration/connector/ha/master/state
```

1.4.5.2.5.2.2 Shadow Instance Configuration

Read and edit the configuration settings for a Cloud Connector shadow instance via API (available as of Cloud Connector version 2.12.0, or, where mentioned, as of version 2.13.0).

> ℹ Note
>
> The APIs below are only permitted on a Cloud Connector shadow instance. The master instance will reject the requests with error code `403 – FORBIDDEN_REQUEST`.

Get Configuration

URI	`/api/v1/configuration/connector/ha/shadow/config`
Method	*GET*
Request	
Response	`{masterHost, masterPort, ownHost, checkIntervalInSeconds, takeoverDelayInSeconds, connectTimeoutInMillis, requestTimeoutInMillis}`
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `masterHost`: Host name of the master instance (string)
- `masterPort`: Port of the master instance (string or number)
- `ownHost`: Host name of the shadow instance (string)
- `checkIntervalInSeconds`: Time between two health checks against the master instance (number)
- `takeoverDelayInSeconds`: The time a master instance may stay unreachable before the shadow instance takes over (number)
- `connectTimeoutInMillis`: Timeout for connection attempts between shadow and master instance (number)
- `requestTimeoutInMillis`: Timeout for requests between shadow and master instance (number)

> **i Note**
>
> This API may take some time to fetch the own hosts from the environment.

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/
configuration/connector/ha/shadow/config
```

Set Configuration

URI	`/api/v1/configuration/connector/ha/shadow/config`
Method	*PUT*
Request	`{masterPort, masterHost, ownHost, checkIntervalInSeconds, takeoverDelayInSeconds, connectTimeoutInMillis, requestTimeoutInMillis}`
Response	`{masterPort, masterHost, ownHost, checkIntervalInSeconds, takeoverDelayInSeconds, connectTimeoutInMillis, requestTimeoutInMillis}`
Errors	
Roles	Administrator

Request Properties:

- `masterHost`: Host name of the master instance (string)
- `masterPort`: Port of the master instance (string or number)
- `ownHost`: Host name of the shadow instance (string)
- `checkIntervalInSeconds`: Time between two health checks against the master instance (number)
- `takeoverDelayInSeconds`: The time a master instance may stay unreachable before the shadow instance takes over (number)
- `connectTimeoutInMillis`: Timeout for connection attempts between shadow and master instance (number)
- `requestTimeoutInMillis`: Timeout for requests between shadow and master instance (number)

Response Properties:

See *Request Properties*.

Example

```
curl -i -k -u <user>:<password> -X PUT https://<host>:<port>/api/v1/
configuration/connector/ha/shadow/config
  -H 'Content-Type: application/json' -d '{"masterHost":"localhost",
"masterPort":"8443", "ownHost":"localhost",
      "checkIntervalInSeconds":30, "takeoverDelayInSeconds":10,
"connectTimeoutInMillis":1000,
      "requestTimeoutInMillis":12000}'
```

Get State

> **i Note**
>
> Available as of version 2.13.0.

URI	`/api/v1/configuration/connector/ha/shadow/state`
Method	*GET*
Request	
Response	`{state, ownHosts, stateMessage, masterVersions}`
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `state`: Possible string values are: INITIAL, DISCONNECTED, DISCONNECTING, HANDSHAKE, INITSYNC, READY, or LOST.
- `ownHosts`: List of alternative host names for the shadow instance.
- `stateMessage`: Message providing details on the current state. This property may not always be present. Typically, this property is available if an error occurred (for example, a failed attempt to connect to the master instance).
- `masterVersions`: Overview of relevant component versions of the master system, including a flag (*property ok*) that indicates whether or not there are incompatibility issues because of differing master and shadow versions.

  > **i Note**
  >
  > This property is only available if the shadow instance is connected to the master instance, or if there has been a successful connection to the master system at some point in the past.

Example

```
curl -i -k -u <user>:<password> -X GET https://<host>:<port>/api/v1/
configuration/connector/ha/shadow/state
```

Change State

URI	`/api/v1/configuration/connector/ha/shadow/state`
Method	*POST*
Request	
Response	`{op, user, password}`
Errors	INVALID_REQUEST, ILLEGAL_STATE
Roles	Administrator

Request Properties:

- `op`: String value representing the state change operation. Possible values are CONNECT or DISCONNECT.
- `user`: User for logon to the master instance
- `password`: Password for logon to the master instance

Errors:

- `INVALID_REQUEST` (400): Invalid or missing property values were supplied; this includes wrong user or password
- `ILLEGAL_STATE` (409): The requested operation cannot be executed given the current state of master and shadow instance. This typically means the master instance does not allow high availability.

> **i Note**
>
> The logon credentials are used for initial logon to master instance only. If a shadow instance is disconnected from its master instance, it will reconnect to the (same) master instance using a certificate. Hence, user and password can be omitted when reconnecting.

Example

```
curl -i -k -u <user>:<password> -X POST https://<host>:<port>/api/v1/
configuration/connector/ha/shadow/state
           -H 'Content-Type: application/json' -d '{"op":"CONNECT",
"user":"<user on master>", "password":"<password>"}'
curl -i -k -u <user>:<password> -X POST https://<host>:<port>/api/v1/
configuration/connector/ha/shadow/state
          -H 'Content-Type: application/json' -d '{"op":"DISCONNECT"}'
```

Reset

> **i** Note
>
> Available as of version 2.13.0.

A successful call to this API deletes master host and port, and restores default values for all other settings related to a connection to the master.

> **⚠** Caution
>
> Do not perform this call if the shadow is connected to a master.

URI	`/api/v1/configuration/connector/ha/shadow/state`
Method	*DELETE*
Request	
Response	
Errors	ILLEGAL_STATE
Roles	Administrator

Errors:

- `ILLEGAL_STATE` (409): the shadow is currently connected to a master.

Example

```
curl -i -k -u <user>:<password> -X DELETE https://<host>:<port>/api/v1/
configuration/connector/ha/shadow/state
```

1.4.5.2.5.3 Proxy Settings

Read and edit the Cloud Connector's proxy settings via API.

Get Proxy Settings

URI	`/api/v1/configuration/connector/proxy`
Method	*GET*
Request	
Response	`{host, port, user}`
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `host`: the name of the proxy host (a string)
- `port`: the port of the proxy host (a string)
- `user`: the user name (a string)

Sample Code

```
curl -ik -u Administrator:<password> https://localhost:8443/api/v1/
configuration/connector/proxy
```

Set Proxy Settings (Master Only)

URI	`/api/v1/configuration/connector/proxy`
Method	*PUT*
Request	`{host, port, user, password}`
Response	
Errors	`INVALID_REQUEST`, `FORBIDDEN_REQUEST`
Roles	Administrator

Request Properties:

- `host`: the name of the proxy host (a string)
- `port`: the port of the proxy host (a string)

- `user`: the user name (a string)
- `password`: the password (a string - optional)

Errors:

- `INVALID_REQUEST` (400): invalid values were supplied, or mandatory values are missing.
- `FORBIDDEN_REQUEST` (403): the target of the call is a shadow instance.

The following example sets empty `user` and `password`.

Sample Code

```
curl -ik -u Administrator:<password> https://localhost:8443/api/v1/
configuration/connector/proxy -X PUT -H 'Content-Type: application/json' -d
'{"host":"proxy", "port":"8080"}'
```

This request removes the proxy configuration.

Sample Code

```
curl -ik -u Administrator:<password> https://localhost:8443/api/v1/
configuration/connector/proxy -X PUT -H 'Content-Type: application/json' -d
'{}'
```

Remove Proxy Settings (Master Only)

URI	`/api/v1/configuration/connector/proxy`
Method	*DELETE*
Request	
Response	204 on success
Errors	`FORBIDDEN_REQUEST`
Roles	Administrator

Errors:

- `FORBIDDEN_REQUEST` (403): the target of the call is a shadow instance.

Sample Code

```
curl -ik -u Administrator:<password> https://localhost:8443/api/v1/
configuration/connector/proxy -X DELETE
```

1.4.5.2.5.4 Authentication and UI Settings

Read and edit the Cloud Connector's authentication and UI settings via API.

Get Authentication Settings

URI	`/api/v1/configuration/connector/authentication`
Method	*GET*
Request	
Response	`{type, configuration}`
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `type`: The authentication type, which is one of the following strings: *basic* or *ldap*.
- `configuration`: The configuration of the active LDAP authentication. This property is only available if `type` is ldap. Its value is an object with properties that provide details on LDAP configuration.

Example

```
curl -i -k -H 'Accept:application/json'
-u Administrator:<password> -X GET https://<scchost>:8443/api/v1/configuration/
connector/authentication
```

Change Basic Authentication User

URI	`/api/v1/configuration/connector/authentication/basic`
Method	*PUT*
Request	`{password, user}`
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `password`: The current password (a string)
- `user`: The new user name (a string), overwriting the current user name.

Errors:

- `INVALID_REQUEST` (400): Current password is wrong.

Change Basic Authentication Password

URI	`/api/v1/configuration/connector/authentication/basic`
Method	*PUT*
Request	`{oldPassword, newPassword}`
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `oldPassword`: The current password, about to be changed (a string)
- `newPassword`: The new password (a string)

Errors:

- `INVALID_REQUEST` (400): Passwords are the same or current password is wrong.

Example

```
curl -i -k -H 'Content-Type:application/json' -d '{"oldPassword"="manage",
"newPassword"="test"}'
-u Administrator:manage -X PUT https://localhost:8443/api/v1/configuration/
connector/authentication/basic
```

Change LDAP Authentication

> ⚠ Caution
>
> The Cloud Connector will restart if the request was successful. There is no test that confirms login will work afterwards. If you run into problems you can revert to basic authentication by executing the script `useFileUserStore` located in the root directory of your Cloud Connector installation.

URI	`/api/v1/configuration/connector/authentication/ldap`
Method	*PUT*
Request	`{enable, configuration}`
Response	204 on success
Errors	INVALID_REQUEST, INVALID_CONFIGURATION
Roles	Administrator

Request Properties:

- `enable`: Boolean flag that indicates whether or not to employ LDAP authentication.
- `configuration`: The LDAP configuration, a JSON object with the properties `{config, hosts, user, password, customAdminRole , customDisplayRole, customMonitoringRole , customSupportRole}`.
  - Property `hosts` is an array. Each element of the array defines a host, again specified through a JSON object, with the properties `{host, port, isSecure}`, accepting string, string (or number), and Boolean values, respectively.
  - All properties of the top-level object except `hosts` accept string values.
  - Properties `config` and `hosts` are mandatory. The array of hosts needs to have at least one element.
  - All other properties are optional.

Errors:

- `INVALID_REQUEST` (400): Configuration is invalid.

- `INVALID_CONFIGURATION` (409): LDAP server is not accessible (does not respond).

> **i Note**
>
> In both error cases nothing is stored, and LDAP authentication is disabled.

Example

```
curl -i -k -H 'Content-Type: application/json' -u Administrator:<password> -X
PUT https://localhost:8443/api/v1/configuration/connector/authentication/ldap -d
'{"enable":true, "configuration":{"hosts":[{"host":"ldaphost", "port":"10389",
"isSecure":false}], "config":"roleBase=\"ou=groups,dc=scc\" roleName=\"cn\"
roleSearch=\"(uniqueMember={0})\" userBase=\"ou=users,dc=scc\" userSearch=
\"(uid={0})\"", "user":"ldapadmin", "password":"<ldapadminpassword>"}}'
```

Get Description for UI Certificate

This API returns a textual description for the system certificate.

> **i Note**
>
> Available as of version 2.13.0.

URI	`/api/v1/configuration/connector/ui/uiCertificate`
Method	*GET*
Response	`{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}`
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `subjectDN`: The subject distinguished name (a string)
- `issuer`: The issuer (a string)
- `notBeforeTimeStamp`: Timestamp of the beginning of the validity period (a UTC long number)
- `notAfterTimeStamp`: Timestamp of the end of the validity period (a UTC long number)

- `subjectAltNames`: Subject alternative names, as an array of objects with properties `type` and `value`. The value of property `type` is one of the following strings: IP, DNS, URI, or RFC822. The value of property `value` is the associated value (a string).

> **Note**
>
> `subjectAltNames` is not present if there are no subject alternative names.

Example

```
curl -i -k -H "Accept: application/json" -u <user>:<password> -X GET https://
<host>:<port>/api/v1/configuration/connector/ui/uiCertificate
```

Create a Self-Signed UI Certificate

> **Note**
>
> Available as of version 2.13.0.

URI	`/api/v1/configuration/connector/ui/uiCertificate`
Method	*POST*
Request	`{type, subjectDN, subjectAltNames}`
Response	201 on success
Errors	
Roles	Administrator

Request Properties:

- `type`: The string *selfsigned*
- `subjectDN`: The subject distinguished name (a string)
- `subjectAltNames`: Subject alternative names, as an array of objects with properties `type` and `value`. The value of property `type` is one of the following strings: IP, DNS, URI, or RFC822. The value of property `value` is the associated value (a string). This property is optional.

The UI certificate created this way has a validity of 1 year.

Example

```
curl -k -u Administrator:<password> -X POST -H  "Content-Type: application/json"
--data  '{"type":"selfsigned", "subjectDN":"CN=me"}'
https: //<host>:<port>/api/v1/configuration/connector/ui/uiCertificate
```

Create a Certificate Signing Request for a UI Certificate

> **i Note**
>
> Available as of version 2.13.0.

URI	`/api/v1/configuration/connector/ui/uiCertificate`
Method	*POST*
Request	`{type, subjectDN, subjectAltNames}`
Response	PEM-encoded certificate request
Errors	
Roles	Administrator

Request Properties:

- `type`: The string *csr*
- `subjectDN`: The subject distinguished name (a string)
- `subjectAltNames`: Subject alternative names, as an array of objects with properties `type` and `value`. The value of property `type` is one of the following strings: IP, DNS, URI, or RFC822. The value of property `value` is the associated value (a string). This property is optional.

Example

```
curl -k -u Administrator:<password> -X POST -H  "Content-Type: application/json"
--data  '{"type":"csr", "subjectDN":"CN=me"}'
https: //<host>:<port>/api/v1/configuration/connector/ui/uiCertificate  -o
csr.pem
```

Upload a Signed Certificate Chain as UI Certificate

> **i Note**
>
> Available as of version 2.13.0.

URI	`/api/v1/configuration/connector/ui/uiCertificate`
Method	*PATCH*
Request	multipart/form-data with the following form parameter: `signedCertificate`
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- `signedCertificate`: The signed certificate and CA certificate chain (PEM-encoded)

Errors:

- `INVALID_REQUEST` (400): The certificate chain provided does not match the most recent certificate request, or it is not a certificate chain in the proper format (PEM-encoded).

Example

```
curl -i -k -u <user>:<password> -X PATCH -F signedCertificate=@<signedchain.pem>
https://<host>:<port>/api/v1/configuration/connector/ui/uiCertificate
```

Example

For test purposes, you can sign the certificate signing request with keytool.

```
keytool -genkeypair -keyalg RSA -keysize  1024 -alias mykey -dname  "cn=very
trusted, c=test" -validity  365 -keystore ca.ks -keypass testit -storepass testit
keytool -gencert -rfc -infile csr.pem -outfile signedcsr.pem -alias mykey -
keystore ca.ks -keypass testit -storepass testit
keytool -exportcert -rfc -file ca.pem -alias mykey -keystore ca.ks -keypass
testit -storepass testit
```

```
cat signedcsr.pem ca.pem > signedchain.pem
```

Upload a PKCS#12 Certificate as UI Certificate

> **i Note**
>
> Available as of version 2.13.0.

URI	`/api/v1/configuration/connector/ui/uiCertificat`
Method	*PUT*

Request	multipart/form-data with the following form parameters:
	`pkcs12`
	`password`
	`keyPassword`

Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- `pkcs12`: Contents of PKCS#12 file
- `password`: Password for decrypting the PKCS#12 file
- `keyPassword`: Optional password for the private key

Errors:

- `INVALID_REQUEST` (400): Contents of PKCS#12 or password are invalid

> **i Note**
>
> `keyPassword` is optional. If missing, `password` is used to decrypt the pkcs#12 file and the private key.

Example

```
curl -i -k -u <user>:<password> https://<host>:<port>/api/v1/configuration/
connector/ui/uiCertificate -X PUT -F 'password=<p12Password>' -F
pkcs12=@<p12file>
```

Example

For test purposes, you can create an own self-signed pkcs#12 certificate with keytool.

```
keytool -genkeypair -alias key -keyalg RSA -keysize 2048 -validity 365 -keypass
test20 -keystore test.p12 -storepass test20 -storetype PKCS12 -dname 'CN=test'
```

1.4.5.2.5.5 Certificate Management for Backend Communication

Manage a CA certificate for principal propagation or a system certificate via API.

> **i Note**
>
> The APIs below are available as of Cloud Connector version 2.13.

There are two similar sets of APIs for system certificate and CA certificate for principal propagation.

> **i Note**
>
> Some of the APIs list a parameter `subjectAltNames` (*subject alternative names* or SAN) for the request or response object. This parameter is an array of objects with the following properties:
>
> - `type`: one of the strings *DNS*, *URI*, *IP*, or *RFC822*.
> - `value`: the value associated with the chosen type.

- CA Certificate for Principal Propagation: APIs [page 383]
- System Certificate: APIs [page 389]

1.4.5.2.5.5.1 CA Certificate for Principal Propagation: APIs

Manage a CA certificate for principal propagation via API.

> **i Note**
>
> The APIs below are available as of Cloud Connector version 2.13.0.

Get Description for a CA Certificate for Principal Propagation

URI	`/api/v1/configuration/connector/onPremise/ppCaCertificate`
Method	*GET*
Header	Accept: application/json
Response	`{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}`
Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response Properties:

- `subjectDN`: the subject distinguished name (a string)
- `issuer`: the issuer (a string)
- `notBeforeTimeStamp`: timestamp of the beginning of the validity period (a UTC long number)
- `notAfterTimeStamp`: timestamp of the end of the validity period (a UTC long number)
- `subjectAltNames`: subject alternative names (see Certificate Management for Backend Communication [page 383] for details).

Errors:

- `NOT_FOUND` (404): there is no CA certificate for principal propagation.

> **i Note**
>
> `subjectAltNames` is not present if there are no subject alternative names.

Example

```
curl -i -k -H "Accept: application/json" -u <user>:<password> -X GET  https://
<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate
```

Get Binary Content of a CA Certificate for Principal Propagation

URI	`/api/v1/configuration/connector/onPremise/ppCaCertificate`
Method	*GET*
Header	Accept: application/pkix-cert
Response	Binary data of the certificate.
Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response:

- `Success`: the binary data of the certificate; you can verify the downloaded certificate by storing it in file `ppca.crt`, for instance, and then running

  ```
  keytool -printcert -file ppca.crt
  ```

- `Failure`: an error in the usual JSON format; the content type of the response is *application/json* in this case.

Errors:

- `NOT_FOUND` (404): there is no CA certificate for principal propagation.

Example

```
curl -k -H "Accept: application/pkix-cert" -u <user>:<password> -X GET --output
sys.crt https://<host>:<port>/api/v1/configuration/connector/onPremise/
ppCaCertificate
```

Create a Self-Signed CA Certificate for Principal Propagation (Master Only)

URI	`/api/v1/configuration/connector/onPremise/ppCaCertificate`

Method	*POST*
Request	`{type, subjectDN, subjectAltNames}`
Response	201 on success
Errors	
Roles	Administrator

Request Properties:

- `type`: the string *selfsigned*
- `subjectDN`: the subject distinguished name (a string)
- `subjectAltNames`: subject alternative names (see Certificate Management for Backend Communication [page 383] for details). This property is optional.

The certificate created this way has a validity of 1 year.

Example

```
curl -i -k -H "Accept: application/json" -H "Content-Type: application/json" -u
<user>:<password> -X POST --data '{"type":"selfsigned", "subjectDN":"CN=me"}'
https://<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate
```

Create a Certificate Signing Request for a CA Certificate for Principal Propagation (Master Only)

URI	`/api/v1/configuration/connector/onPremise/ppCaCertificate`
Method	*POST*
Request	`{type, subjectDN, subjectAltNames}`
Response	PEM-encoded certificate request
Errors	
Roles	Administrator

Request Properties:

- `type`: the string *selfsigned*
- `subjectDN`: the subject distinguished name (a string)
- `subjectAltNames`: subject alternative names (see Certificate Management for Backend Communication [page 383] for details). This property is optional.

Example

```
curl -k -u <user>:<password> -X POST -H "Content-Type: application/json" --data
'{"type":"csr", "subjectDN":"CN=me"}'
https://<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate  -
o csr.pem
```

Upload a Signed Certificate Chain as CA Certificate for Principal Propagation (Master Only)

URI	`/api/v1/configuration/connector/onPremise/ppCaCertificate`
Method	*PATCH*
Request	multipart/form-data with the following parameter: `signedCertificate`.
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `signedCertificate`: the signed certificate and CA certificate chain (PEM-encoded)

Errors:

- `INVALID_REQUEST` (400): the certificate chain provided does not match the most recent certificate request, or it is not a certificate chain in the correct format (PEM-encoded).

Example

```
curl -i -k -u <user>:<password> -X PATCH -F signedCertificate=@<signedchain.pem>
https://<host>:<port>/api/v1/configuration/connector/onPremise/ppCaCertificate
```

Example: Sign The Certificate Signing Request

```
keytool -genkeypair -keyalg RSA -keysize 1024 -alias mykey -dname "cn=very
trusted, c=test" -validity 365 -keystore ca.ks -keypass testit -storepass testit
keytool -gencert -rfc -infile csr.pem -outfile signedcsr.pem -alias mykey -
keystore ca.ks -keypass testit -storepass testit
keytool -exportcert -rfc -file ca.pem -alias mykey -keystore ca.ks -keypass
testit -storepass testit
cat signedcsr.pem ca.pem > signedchain.pem
```

Upload a PKCS#12 Certificate as CA Certificate for Principal Propagation (Master Only)

URI	`/api/v1/configuration/connector/onPremise/ppCaCertificate`
Method	*PUT*
Request	Multipart/form-data with the following form parameters: `pkcs12`, `password`, `keyPassword`
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- `pkcs12`: contents of PKCS#12 file
- `password`: password for decrypting PKCS#12 file
- `keyPassword`: optional password for the private key

Errors:

- `INVALID_REQUEST` (400): contents of PKCS#12 or password are invalid

> **i Note**
>
> `keyPassword` is optional. If it is missing, `password` is used to decrypt the pkcs#12 file and the private key.

Example

```
curl -i -k -u <user>:<password> https://<host>:<port>/api/v1/configuration/
connector/onPremise/ppCaCertificate -X PUT -F 'password=<p21Password>' -F
pkcs12=@<p12file>
```

Create an own self-signed pkcs#12 certificate for tests with:

```
keytool -genkeypair -alias key -keyalg RSA -keysize 2048 -validity 365 -keypass
test20 -keystore test.p12 -storepass test20 -storetype PKCS12 -dname 'CN=test'
```

Delete a CA Certificate for Principal Propagation (Master Only)

URI	`/api/v1/configuration/connector/onPremise/ppCaCertificate`
Method	*DELETE*
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator

Errors:

- `NOT_FOUND` (404): there is no CA certificate for principal propagation.

Example

```
curl -k -H "Accept: application/json" -u <user>:<password> --request DELETE
https://localhost:8443/api/v1/configuration/connector/onPremise/ppCaCertificate
```

1.4.5.2.5.5.2 System Certificate: APIs

Manage a system certificate via API.

> **i Note**
>
> The APIs below are available as of Cloud Connector version 2.13.0.

Get Description for a System Certificate

URI	`/api/v1/configuration/connector/onPremise/systemCertificate`
Method	*GET*
Header	Accept: application/json
Response	`{subjectDN, issuer, notBeforeTimeStamp, notAfterTimeStamp, subjectAltNames}`
Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response Properties:

- `subjectDN`: the subject distinguished name (a string)
- `issuer`: the issuer (a string)
- `notBeforeTimeStamp`: timestamp of the beginning of the validity period (a UTC long number)
- `notAfterTimeStamp`: timestamp of the end of the validity period (a UTC long number)
- `subjectAltNames`: subject alternative names (see Certificate Management for Backend Communication [page 383] for details).

Errors:

- `NOT_FOUND` (404): there is no system certificate.

> **i Note**
>
> `subjectAltNames` is not present if there are no subject alternative names.

Example

```
curl -i -k -H "Accept: application/json" -u <user>:<password> -X GET  https://
<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate
```

Get Binary Content of a System Certificate

URI	`/api/v1/configuration/connector/onPremise/systemCertificate`
Method	*GET*
Header	Accept: application/pkix-cert
Response	Binary data of the certificate.
Errors	NOT_FOUND
Roles	Administrator, Display, Support

Response:

- `Success`: the binary data of the certificate; you can verify the downloaded certificate by storing it in file `sys.crt`, for instance, and then running

  ```
  keytool -printcert -file sys.crt
  ```

- `Failure`: an error in the usual JSON format; the content type of the response is *application/json* in this case.

Errors:

- `NOT_FOUND` (404): there is no system certificate.

Example

```
curl -i -k -H "Accept: application/pkix-cert" -u <user>:<password> -X GET --
output sys.crt https://<host>:<port>/api/v1/configuration/connector/onPremise/
systemCertificate
```

Create a Self-Signed System Certificate (Master Only)

URI	`/api/v1/configuration/connector/onPremise/systemCertificate`
Method	*POST*
Header	CONTENT_TYPE: application/json

Request	`{type, subjectDN, subjectAltNames}`
Response	201 on success
Errors	
Roles	Administrator

Request Properties:

- `type`: the string *selfsigned*
- `subjectDN`: the subject distinguished name (a string)
- `subjectAltNames`: subject alternative names (see Certificate Management for Backend Communication [page 383] for details). This property is optional.

The certificate created this way has a validity of 1 year.

Example

```
curl -i -k -H "Accept: application/json" -H "Content-Type: application/json" -u
<user>:<password> -X POST --data '{"type":"selfsigned", "subjectDN":"CN=me"}'
https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate
```

Create a Certificate Signing Request for a System Certificate (Master Only)

URI	`/api/v1/configuration/connector/onPremise/systemCertificate`
Method	*POST*
Header	CONTENT_TYPE: application/json
Request	`{type, subjectDN, subjectAltNames}`
Response	PEM-encoded certificate request
Errors	
Roles	Administrator

Request Properties:

- `type`: the string *selfsigned*

- `subjectDN`: the subject distinguished name (a string)
- `subjectAltNames`: subject alternative names (see Certificate Management for Backend Communication [page 383] for details). This property is optional.

Example

```
curl -k -u <user>:<password> -X POST -H "Content-Type: application/json" --data
'{"type":"csr", "subjectDN":"CN=me"}'
https://<host>:<port>/api/v1/configuration/connector/onPremise/
systemCertificate  -o csr.pem
```

Upload a Signed Certificate Chain as System Certificate (Master Only)

URI	`/api/v1/configuration/connector/onPremise/systemCertificate`
Method	*PATCH*
Request	multipart/form-data with the following parameter: `signedCertificate`.
Response	201 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Properties:

- `signedCertificate`: the signed certificate and CA certificate chain (PEM-encoded)

Errors:

- `INVALID_REQUEST` (400): the certificate chain provided does not match the most recent certificate request, or it is not a certificate chain in the correct format (PEM-encoded).

Example

```
curl -i -k -u <user>:<password> -X PATCH -F signedCertificate=@<signedchain.pem>
https://<host>:<port>/api/v1/configuration/connector/onPremise/systemCertificate
```

For test purposes, you can sign the certificate signing request with keytool:

```
keytool -genkeypair -keyalg RSA -keysize 1024 -alias mykey -dname "cn=very
trusted, c=test" -validity 365 -keystore ca.ks -keypass testit -storepass testit
keytool -gencert -rfc -infile csr.pem -outfile signedcsr.pem -alias mykey -
keystore ca.ks -keypass testit -storepass testit
keytool -exportcert -rfc -file ca.pem -alias mykey -keystore ca.ks -keypass
testit -storepass testit
cat signedcsr.pem ca.pem > signedchain.pem
```

Upload a PKCS#12 Certificate as System Certificate (Master Only)

URI	`/api/v1/configuration/connector/onPremise/systemCertificate`
Method	*PUT*
Request	Multipart/form-data with the following form parameters: `pkcs12`, `password`, `keyPassword`
Response	204 on success
Errors	INVALID_REQUEST
Roles	Administrator

Request Parameters:

- `pkcs12`: contents of PKCS#12 file
- `password`: password for decrypting PKCS#12 file
- `keyPassword`: optional password for the private key

Errors:

- `INVALID_REQUEST` (400): contents of PKCS#12 or password are invalid

> **i Note**
>
> `keyPassword` is optional. If it is missing, `password` is used to decrypt the pkcs#12 file and the private key.

Example

```
curl -i -k -u <user>:<password> https://<host>:<port>/api/v1/configuration/
connector/onPremise/systemCertificate -X PUT -F password=<p21Password> -F
pkcs12=@<p12file>
```

Create an own self-signed pkcs#12 certificate for tests with:

```
keytool -genkeypair -alias key -keyalg RSA -keysize 2048 -validity 365 -keypass
test20 -keystore test.p12 -storepass test20 -storetype PKCS12 -dname 'CN=test'
```

Delete a System Certificate (Master Only)

URI	`/api/v1/configuration/connector/onPremise/systemCertificate`
Method	*DELETE*
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator

Errors:

- `NOT_FOUND` (404): there is no system certificate.

Example

```
curl -k -H "Accept: application/json" -u <user>:<password> --request DELETE
https://localhost:8443/api/v1/configuration/connector/onPremise/systemCertificate
```

1.4.5.2.5.6 Solution Management Configuration

Manage the Cloud Connector's solution management configuration via API.

Get Solution Management Configuration

URI	`/api/v1/configuration/connector/solutionManagement`
Method	*GET*
Request	
Response	`{hostAgentPath, isEnabled, dsrEnabled}`
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `isEnabled`: flag indicating if reporting to solution management is active.
- `hostAgentPath`: path for host agent executable.
- `dsrEnabled`: indicating if DSR reporting is active.

Example

```
curl -ik -u <user>:<password>  https://<host>:<port>/api/v1/configuration/
connector/solutionManagement
```

Set Solution Management Configuration and Turn On Reporting

This API turns on the integration with the Solution Manager. The prerequisite is an available Host Agent. You can specify a path to the Host Agent executable, if you don't use the default path.

URI	`/api/v1/configuration/connector/solutionManagement`

Method	*POST*
Request	`{hostAgentPath, dsrEnabled}`
Response	
Errors	
Roles	Administrator, Support

Response Properties:

- `hostAgentPath`: path for host agent executable (string, optional).
- `dsrEnabled`: flag indicating if DSR reporting is active (boolean, optional)-

Example

```
curl -ik -u <user>:<password>  https://<host>:<port>/api/v1/configuration/
connector/solutionManagement -X POST
```

or, if configuration has to be changed:

```
curl -ik -u <user>:<password>  https://<host>:<port>/api/v1/configuration/
connector/solutionManagement -X POST -d "{\"hostAgentPath\":\"new/path/to/
hostAgent\"}" -H "Content-Type:application/json"
```

Turn Off Solution Management Reporting

URI	`/api/v1/configuration/connector/solutionManagement`
Method	*DELETE*
Request	
Response	
Errors	
Roles	Administrator, Support

Example

```
curl -ik -u <user>:<password>  https://<host>:<port>/api/v1/configuration/
connector/solutionManagement -X DELETE
```

Download Current LMDB XML Report

Generates a zip file containing the registration file for the solution management LMDB (Landscape Management Database).

> **i** Note
>
> Available as of Cloud Connector version 2.12.0.

URI	`/api/v1/configuration/connector/solutionManagement/registrationFile`
Method	*GET*
Request	
Response	
Errors	
Roles	Administrator, Support

1.4.5.2.5.7 Backup

Manage the Cloud Connector's configuration backup via API.

Create Backup Configuration

URI	`/api/v1/configuration/backup`
Method	*POST*

Request	`{password}`
Response	ZIP archive (content type `application/zip`)
Errors	
Roles	Administrator

Request Properties:

- `password`: the password used to encrypt sensitive data.

> **i Note**
>
> Only sensitive data in the backup are encrypted with an arbitrary password of your choice. The password is required for the restore operation. The returned ZIP archive itself is not password-protected.

Restore Backup Configuration

URI	`/api/v1/configuration/backup`
Method	*PUT*
Request	Multipart/form-data backup file and password as form parameter. Successful request forces restart of the Cloud Connector.
Response	204 - in case of success.
Errors	400 `{type: INVALID_REQUEST, message: <msg>}`
Roles	Administrator

> **i Note**
>
> Since this API uses a multipart request, it requires a multipart request header.

1.4.5.2.5.8 Subaccount

Manage the Cloud Connector's subaccount settings via API.

Operations

Subaccount	Get Subaccounts [page 400] Create Subaccount (Master Only) [page 401] Delete Subaccount (Master Only) [page 402] Edit Subaccount (Master Only) [page 402] Connect/Disconnect Subaccount (Master Only) [page 403] Refresh Subaccount Certificate (Master Only) [page 404] Get Subaccount Configuration [page 405]
Recovery Subaccount	Create Recovery Subaccount (Master Only) [page 405] Delete Recovery Subaccount (Master Only) [page 406] Refresh Certificate of Recovery Subaccount (Master Only) [page 407] Activate/Deactivate Recovery Subaccount (Master Only) [page 408] Takeover (Master Only) [page 408]

Get Subaccounts

URI	`/api/v1/configuration/subaccounts`
Method	*GET*
Request	
Response	`[{regionHost, subaccount, locationID}]`
Errors	
Roles	Administrator, Display, Support

Response:

An array of objects with the following properties:

- `regionHost`: region hosts (a string).
- `subaccount`: subaccount name (a string).
- `locationID`: location identifier for the Cloud Connector instance (a string); this property is not available if the default location ID is in use.

Back to Operations [page 400]

Create Subaccount (Master Only)

Creates and connects a subaccount.

URI	`/api/v1/configuration/subaccounts`
Method	*POST*
Request	`{regionHost, subaccount, cloudUser, cloudPassword, locationID, displayName, description}`
Response	201, created subaccount entity: `{regionHost, subaccount, locationID, displayName, description, tunnel}`
Errors	INVALID_REQUEST, INVALID_CONFIGURATION
Roles	Administrator

Request Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).
- `cloudUser`: user for the specified subaccount and region host.
- `cloudPassword`: password for the cloud user.
- `locationID`: location identifier for the Cloud Connector instance (a string; optional).
- `displayName`: display name of the subaccount (a string; optional).
- `description`: subaccount description (a string; optional).

Response Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).
- `locationID`: location ID (a string); this property is not available if the default location ID is in use.

- `displayName`: display name of the subaccount (a string); this property is not available if there is no specified display name.
- `description`: subaccount description (a string); this property is not available if there is no description.
- `tunnel`: object outlining the current state of the tunnel.

Errors:

- INVALID_REQUEST (400): one or more mandatory parameters are missing or invalid.
- INVALID_CONFIGURATION (409): the subaccount already exists as a regular subaccount or recovery subaccount.

Back to Operations [page 400]

Delete Subaccount (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>`
Method	*DELETE*
Request	
Response	204 on success
Errors	NOT_FOUND, ILLEGAL_STATE
Roles	Administrator

Errors:

- `NOT_FOUND` (404): subaccount does not exist (in the specified region).
- `ILLEGAL_STATE` (409): there is at least one session that has access to the subaccount.

Back to Operations [page 400]

Edit Subaccount (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>`
Method	*PUT*

Request	`{locationID, displayName, description}`
Response	`{regionHost, subaccount, locationID, displayName, description, tunnel, recoveryAccountState}`
Errors	NOT_FOUND
Roles	Administrator

Request Properties:

- `locationID`: location identifier for the Cloud Connector instance (a string; optional); if this parameter is not supplied the location ID will not change. Revert to the default location ID by supplying the empty string.
- `displayName`: subaccount display name (a string; optional); if this parameter is not supplied the display name will not change. Clear the display name by using an empty string.
- `description`: subaccount description (a string; optional); if this parameter is not supplied the description will not change. Clear the description by using an empty string.

Response Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).
- `locationID`: location identifier for the Cloud Connector instance (a string); this property is not available if the default location ID is in use.
- `displayName`: display name of the subaccount (a string); this property is not available if there is no specified display name.
- `description`: subaccount description (a string); this property is not available if there is no description.
- `tunnel`: object outlining the current state of the tunnel.
- `recoveryAccountState`: object detailing the current state of the recovery subaccount. This property is supplied only if a recovery subaccount is configured.

Errors

- `NOT_FOUND` (404): subaccount does not exist (in the specified region).

Back to Operations [page 400]

Connect/Disconnect Subaccount (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/state`

Method	*PUT*
Request	`{connected}`
Response	204 on success
Errors	
Roles	Administrator

Request Properties:

- `connected`: a Boolean value indicating whether the subaccount should be connected (true) or disconnected (false).

Back to Operations [page 400]

Refresh Subaccount Certificate (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/validity`
Method	*POST*
Request	`{user, password}`
Response	`{regionHost, subaccount, locationID, displayName, description, tunnel, recoveryAccountState}`
Errors	
Roles	Administrator

Request Properties:

- `user`: user for the specified region host and subaccount.
- `password`: password for the (cloud) user.

Response Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).
- `locationID`: location identifier for the Cloud Connector instance (a string); this property is not available if the default location ID is in use.

- `displayName`: display name of the subaccount (a string); this property is not available if there is no specified display name.
- `description`: subaccount description (a string); this property is not available if there is no description.
- `tunnel`: object outlining the current state of the tunnel.
- `recoveryAccountState`: object detailing the current state of the recovery subaccount. This property is supplied only if a recovery subaccount is configured.

Back to Operations [page 400]

Get Subaccount Configuration

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>`
Method	*GET*
Request	
Response	`{regionHost, subaccount, locationID, displayName, description, tunnel: {state, connections, applicationConnections:[], serviceChannels:[]}}`
Errors	
Roles	Administrator, Display, Support

Response Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).
- `locationID`: location ID (a string); this property is not available if the default location ID is in use.
- `displayName`: display name of the subaccount (a string); this property is not available if there is no specified display name.
- `description`: description (a string); this property is not available if there is no description.
- `tunnel`: object outlining the current state of the tunnel.

Back to Operations [page 400]

Create Recovery Subaccount (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery`
Method	*POST*
Request	`{regionHost, cloudUser, cloudPassword}`
Response	201 on success
Errors	INVALID_REQUEST, ILLEGAL_STATE
Roles	Administrator

Request Properties:

- `regionHost`: region host of the recovery subaccount.
- `cloudUser`: user for the specified region host and recovery subaccount.
- `cloudPassword`: password for the cloud user.

Errors:

- `INVALID_REQUEST` (400): invalid or missing request parameters.
- `ILLEGAL_STATE` (409): a recovery subaccount already exists.

> **i Note**
>
> A recovery subaccount cannot be changed. Delete it and create a new one instead.

Back to Operations [page 400]

Delete Recovery Subaccount (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery`
Method	*DELETE*
Request	
Response	
Errors	NOT_FOUND
Roles	Administrator

Errors:

- `NOT_FOUND` (404): there is no recovery subaccount.

Back to Operations [page 400]

Refresh Certificate of Recovery Subaccount (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/validity`
Method	*POST*
Request	`{user, password}`
Response	`{regionHost, subaccount, locationID, displayName, description, tunnel, recoveryAccountState}`
Errors	INVALID_REQUEST, NOT_FOUND
Roles	Administrator

Request Properties:

- `user`: user for the specified region host and subaccount.
- `password`: password for the (cloud) user.

Response Properties:

- `regionHost`: region host name (a string).
- `subaccount`: subaccount technical name (a string).
- `locationID`: location identifier for the Cloud Connector instance (a string); this property is not available if the default location ID is in use.
- `displayName`: display name of the subaccount (a string); this property is not available if there is no specified display name.
- `description`: subaccount description (a string); this property is not available if there is no description.
- `tunnel`: object outlining the current state of the tunnel.
- `recoveryAccountState`: object detailing the current state of the recovery subaccount. This property is supplied only if a recovery subaccount is configured.

Errors

- `INVALID_REQUEST` (400): user or password are missing.

- `NOT_FOUND` (404): there is no recovery subaccount.

Back to Operations [page 400]

Activate/Deactivate Recovery Subaccount (Master Only)

> **i Note**
>
> Available as of Cloud Connector 2.13.1.

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/state`
Method	*PUT*
Request	`{active}`
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator

Request Properties:

- `active`: Boolean value indicating whether the recovery subaccount should be active (true) or inactive (false).

Errors:

- `NOT_FOUND` (404): there is no recovery subaccount.

Back to Operations [page 400]

Takeover (Master Only)

> **⚠ Caution**
>
> When performing this operation, the recovery subaccount *permanently takes over* from the original subaccount, and the *original subaccount is deleted*. The *recovery subaccount must be active* for takeover to succeed, see Activate/Deactivate Recovery Subaccount (Master Only) [page 408].

> **i Note**
>
> Available as of Cloud Connector 2.13.1.

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/recovery/takeover`
Method	*POST*
Request	
Response	204 on success
Errors	NOT_FOUND, ILLEGAL_STATE
Roles	Administrator

Errors:

- `NOT_FOUND` (404): there is no recovery subaccount.
- `ILLEGAL_STATE` (409): recovery subaccount is not active.

Back to Operations [page 400]

1.4.5.2.5.9 System Mappings

Manage the Cloud Connector's system mappings via API.

Get All System Mappings

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings`
Method	*GET*
Request	

Response	`[{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader, description, ...}]`
Errors	
Roles	Administrator, Display, Support

Response:

An array of objects with the following properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)
- `localHost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses (a string)
- `backendType`: Type of the backend (a string)
- `authenticationMode`: Authentication mode used on the backend side (a string).
- `hostInHeader`: Policy for setting the host in the response header. Available for HTTP(S) protocols only.
- `totalResourcesCount`: the total number of resources
- `enabledResourcesCount`: the number of enabled resources
- `description`: Description for the system mapping (a string).
- `sncPartnerName`: SNC name of an ABAP Server, required for RFCS communication only.
- `sapRouter`: SAP router route, required only if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedClientUsers`: Array of `{client, user}`, describing users that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

Get System Mapping

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>`
Method	*GET*
Request	

Response	`{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader, description, ...}`
Errors	
Roles	Administrator, Display, Support

Response:

An array of objects with the following properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)
- `localHost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses (a string)
- `backendType`: Type of the backend (a string)
- `authenticationMode`: Authentication mode used on the backend side (a string).
- `hostInHeader`: Policy for setting the host in the response header (a string). Available for HTTP(S) protocols only.
- `totalResourcesCount`: the total number of resources
- `enabledResourcesCount`: the number of enabled resources
- `description`: Description for the system mapping (a string).
- `sncPartnerName`: SNC name of an ABAP Server, required for RFCS communication only.
- `sapRouter`: SAP router route, required only if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedClientUsers`: Array of `{client, user}`, describing users that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

Create System Mapping (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings`
Method	*POST*

Request	`{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader, description, ...}`
Response	201 on success
Errors	
Roles	Administrator

Request Properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)
- `localHost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses, which must be one of the following strings: `HTTP, HTTPS, RFC, RFCS, LDAP, LDAPS, TCP, TCPS`.
- `backendType`: Type of the backend system. Valid values are `abapSys, netweaverCE, netweaverGW, applServerJava, BC, PI, hana, otherSAPsys, nonSAPsys`.
- `authenticationMode`: Authentication mode to be used on the backend side, which must be one of the following strings: `NONE, X509_GENERAL, X509_RESTRICTED, KERBEROS`.
- `hostInHeader`: Policy for setting the host in the response header. This property is applicable to HTTP(S) protocols only, and it is **optional**. If set, it must be one of the following strings: `internal, virtual`. The default is `virtual`. You may also use all capital letters, i.e. `INTERNAL` and `VIRTUAL`.
- `description`: Description for the system mapping (string, optional). The default is no description, i.e. the empty string.
- `sncPartnerName`: SNC name of an ABAP Server, required for RFCS communication only.
- `sapRouter`: SAP router route, required only if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedClientUsers`: Array of `{client, user}`, describing users that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

Delete System Mapping (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings`
Method	*DELETE*

Request	
Response	204 on success
Errors	
Roles	Administrator

Delete All System Mappings (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings`
Method	*DELETE*
Request	
Response	204 on success
Errors	
Roles	Administrator

Edit System Mapping

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/virtualHost:virtualPort`
Method	*PUT*
Request	`{virtualHost, virtualPort, localHost, localPort, protocol, backendType, authenticationMode, hostInHeader, description, ...}`
Response	
Errors	
Roles	Administrator

Request Properties:

- `virtualHost`: Virtual host used on the cloud side (a string)
- `virtualPort`: Virtual port used on the cloud side (a string)

These two properties are technically mandatory as they identify the system mapping to be edited or changed. They cannot be changed. However, they are also part of the URI path. If omitted from the request, they will effectively be copied from the URI path, and no error is thrown. This policy of leniency deviates from strict REST conventions, but was adopted to increase user-friendliness by avoiding unnecessary error situations.

All other properties are optional. Add those properties that you want to change.

- `localHost`: Host on the on-premise side (a string)
- `localPort`: Port on the on-premise side (a string)
- `protocol`: Protocol used when sending requests and receiving responses, which must be one of the following strings: `HTTP, HTTPS, RFC, RFCS, LDAP, LDAPS, TCP, TCPS`.
- `backendType`: Type of the backend system. Valid values are `abapSys, netweaverCE, netweaverGW, applServerJava, BC, PI, hana, otherSAPsys, nonSAPsys`.
- `authenticationMode`: Authentication mode to be used on the backend side, which must be one of the following strings: `NONE, X509_GENERAL, X509_RESTRICTED, KERBEROS`.
- `hostInHeader`: Policy for setting the host in the response header; this property is **optional**. If set, it must be one of the following strings: `internal, virtual`. The default is `virtual`. You may also use all capital letters, i.e. `INTERNAL` and `VIRTUAL`.
- `description`: Description for the system mapping (string, optional). The default is no description, i.e. the empty string.
- `sncPartnerName`: SNC name of an ABAP Server, only set for RFCS communication.
- `sapRouter`: SAP router route, only set if an SAP router is used.
- `allowedClients`: Array of strings, describing the SAP clients allowing to execute the calls in this system. Valid clients are 3 letters long. If no clients are defined here, there is no restriction – every client is allowed. Only applicable for RFC-based communication.
- `blacklistedClientUsers`: Array of `{client, user}`, describing users, that are not allowed to execute the call, even if the client is listed under allowed clients. Only applicable for RFC-based communication.

1.4.5.2.5.10 System Mapping Resources

Manage the Cloud Connector's system mapping resources via API.

Get System Mapping Resources

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/virtualHost:virtualPort/resources`
Method	*GET*
Request	
Response	`[{id, enabled, exactMatchOnly, websocketUpgradeAllowed, description}]`
Errors	
Roles	Administrator, Display, Support

Response:

An array of objects, each representing a resource through the following properties:

- `id`: The resource itself, which, depending on the owning system mapping, is either a URL path (or the leading section of it), or a RFC function name (prefix)
- `enabled`: Boolean flag indicating whether the resource is enabled.
- `exactMatchOnly`: Boolean flag determining whether access is granted only if the requested resource is an exact match.
- `websocketUpgradeAllowed`: Boolean flag indicating whether websocket upgrade is allowed; this property is of relevance only if the owning system mapping employs protocol HTTP or HTTPS.
- `description`: Description (a string); this property is not available unless explicitly set.

Get System Mapping Resource

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/virtualHost:virtualPort/resources/<encodedResourceId>`
Method	*GET*
Request	
Response	`{id, enabled, exactMatchOnly, websocketUpgradeAllowed, description}`
Errors	

Roles	Administrator, Display, Support

Response Properties:

- `id`: The resource itself, which, depending on the owning system mapping, is either a URL path (or the leading section of it), or a RFC function name (prefix)
- `enabled`: Boolean flag indicating whether the resource is enabled.
- `exactMatchOnly`: Boolean flag determining whether access is granted only if the requested resource is an exact match.
- `websocketUpgradeAllowed`: Boolean flag indicating whether websocket upgrade is allowed; this property is of relevance only if the owning system mapping employs protocol HTTP or HTTPS.
- `description`: Description (a string); this property is not available unless explicitly set.

Create System Mapping Resource

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/virtualHost:virtualPort/resources`
Method	*POST*
Request	`{id, enabled, exactMatchOnly, websocketUpgradeAllowed, description}`
Response	201 on success
Errors	
Roles	Administrator

Request Properties:

- `id`: The resource itself, which, depending on the owning system mapping, is either a URL path (or the leading section of it), or a RFC function name (prefix).
- `enabled`: Boolean flag indicating whether the resource is enabled (**optional**). The default value is `false`.
- `exactMatchOnly`: Boolean flag determining whether access is granted only if the requested resource is an exact match (**optional**). The default value is `false`.
- `websocketUpgradeAllowed`: Boolean flag indicating whether websocket upgrade is allowed (**optional**). The default value is `false`. This property is recognized only if the owning system mapping employs protocol HTTP or HTTPS.
- `description`: Description (a string, **optional**)

→ Tip

Encoded Resource ID

URI paths may contain the resource ID in order to identify the resource to be edited or deleted. A resource ID, however, may contain characters such as the forward slash that collide with the path separator of the URI and hence require an escape mechanism. We adopted the following simple escape or encoding method for a resource ID:

1. Replace all occurrences of character '+' with '+2B'.
2. Replace all occurrences of character '-' with '+2D'.
3. Replace all occurrences of character '/' with '-'.

Edit System Mapping Resource

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/virtualHost:virtualPort/resources/<encodedResourceId>`
Method	*PUT*
Request	`{enabled, exactMatchOnly, websocketUpgradeAllowed, description}`
Response	204 on success
Errors	
Roles	Administrator

Request Properties:

- `enabled`: Boolean flag indicating whether the resource is enabled.
- `exactMatchOnly`: Boolean flag determining whether access is granted only if the requested resource is an exact match.
- `websocketUpgradeAllowed`: Boolean flag indicating whether websocket upgrade is allowed; this property is of relevance only if the owning system mapping employs protocol HTTP or HTTPS.
- `description`: Description (a string); this property is not available unless explicitly set.

Delete System Mapping Resource

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/systemMappings/virtualHost:virtualPort/resources/<encodedResourceId>`
Method	*DELETE*
Request	
Response	204 on success
Errors	`NOT_FOUND`
Roles	Administrator

Errors:

`NOT_FOUND (404)`: Resource was not found

Delete All System Mapping Resources

URI	`/api/v1/configuration/subaccounts/<region>/<subaccount>/systemMappings/<virtualHost>:<virtualPort>/resources`
Method	*DELETE*
Request	
Response	204 on success
Errors	
Roles	Administrator

1.4.5.2.5.11 Domain Mappings

Manage the Cloud Connector's configuration for domain mappings via API.

Get Domain Mappings

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/domainMappings`
Method	*GET*
Request	
Response	`[{virtualDomain, internalDomai}]`
Errors	
Roles	Administrator, Display, Support

Response:

An array of objects, each representing a domain mapping through the following properties:

- `virtualDomain`: Domain used on the cloud side
- `internalDomain`: Domain used on the on-premise side

Create Domain Mappings (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/domainMappings`
Method	*POST*
Request	`{virtualDomain, internalDomain}`
Response	201
Errors	
Roles	Administrator

Edit Domain Mappings (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/domainMappings/<internalDomain>`
Method	*PUT*
Request	`{virtualDomain, internalDomain}`
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator

Request:

- `virtualDomain`: New virtual domain
- `internalDomain`: New internal domain

Errors:

- NOT_FOUND (404): Domain mapping does not exist.

> **i Note**
>
> The internal domain in the URI path (i.e., <internalDomain>) is the current internal domain of the domain mapping that is to be edited. It may differ from the new internal domain set in the request.

Delete Domain Mappings (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/domainMappings/<internalDomain>`
Method	*DELETE*
Request	
Response	204 on success
Errors	NOT_FOUND
Roles	Administrator

Errors:

- NOT_FOUND (404): Domain mapping does not exist.

Delete All Domain Mappings (Master Only)

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/domainMappings`
Method	*DELETE*
Request	
Response	204 on success
Errors	
Roles	Administrator

1.4.5.2.5.12 Subaccount Service Channels

Manage Cloud Connector service channels via API.

Get Service Channels

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels`
Method	*GET*
Request	
Response	`[{typeDesc, details, port, enabled, connected, connectionsCount, availableConnectionsCount, connectedSinceTimeStamp}]`
Errors	
Roles	Administrator, Display, Support

Response:

An array of objects, each of which represents a service channel through the following properties:

- `typeDesc`: an object specifying the service channel type through the properties `typeKey` and `typeName`.
- `details`

- `port`
- `enabled`
- `connected`
- `connectionsCount`
- `availableConnectionsCount`
- `connectedSinceTimeStamp`

Create Service Channel

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels`
Method	*POST*
Request	`{typeKey,details,serviceNumber,connectionCount}`
Response	201, Location header for new entity.
Errors	400 - if a parameter is invalid or account or service channel are invalid: `{type: INVALID_REQUEST, message: <msg>}`
Roles	Administrator

Request Properties:

- `typeKey`: type of service channel. Valid values are HANA_DB, HCPVM, RFC.
- `details`:
  - HANA instance name for HANA_DB
  - VM name for HCPVM
  - S/4HANA Cloud tenant host for RFC
- `serviceNumber`: service number, which is mapped to a port according to the type of service channel.
- `connectionCount`: number of connections for the channel.

Delete Service Channel

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/<id>`
Method	*DELETE*
Request	
Response	`{}`
Errors	
Roles	Administrator

Edit Service Channel

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/<id>`
Method	*PUT*
Request	`{typeKey,details,serviceNumber,connectionCount}`
Response	204
Errors	400 - if account or service channel Id are invalid: `{type: INVALID_REQUEST, message: <msg>}`
Roles	Administrator

Enable/Disable Service Channel

URI	`/api/v1/configuration/subaccounts/<regionHost>/<subaccount>/channels/<id>/state`
Method	*PUT*

Request	`{enabled:boolean}`
Response	

Errors	400 - if account or service channel Id are invalid:
	`{type: "INVALID_REQUEST", message: <msg>}`
	500
	`{type:"RUNTIME_FAILURE","message":"Service channel could not be opened"}`

Roles	Administrator

1.4.5.2.5.13 Examples

Find examples on how to use the Cloud Connector's configuration REST APIs.

Concept

The sample code in this section (download zip file) demonstrates how to use the REST APIs provided by Cloud Connector to perform various configuration tasks.

Starting with a freshly installed Cloud Connector, the samples include initial configuration of the Cloud Connector instance, connectivity setup, high availability configuration, and common tasks like backup/restore operations, as well as integration with solution management.

The examples are implemented in *Kotlin*, a simple Java VM-based language. However, even if you are using a different language, they still show the basic use of the APIs and their parameters for specific configuration purposes.

If you are not familiar with Kotlin, find a brief introduction and some typical statements below.

REST API Parameters [page 425]

REST API Invocation [page 425]

How To Use the Examples [page 426]

REST API Parameters

In almost all requests and responses, structures are encoded in JSON format. To describe the parameter details, we use Kotlin data classes.

This class represents a structure that you can use as value in a request or response:

```
data class OnlyPropertiesNamesAreRelevant(
    val user: String,
    val password: String
)
```

The JSON representation for that class is

```
{"user":<userValue>, "password":<passwordValue>}
```

Encoded in Kotlin this string looks like

```
"""{"user":"$userValue", "password":"$passwordValue"}"""
```

or as an object:

```
val credentials = OnlyPropertiesNamesAreRelevant(userValue, passwordValue)
```

Back to Concept [page 424]

REST API Invocation

```
(a)     Fuel.put(url)
(b)      .header("Connection", "close")
(c)         .authentication().basic(user, password)
(d)         .jsonBody(credentials)
(e)         .responseObject<OnlyPropertiesNamesAreRelevant> { _, _, result ->
             when (result) {
(f)                 is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("returned: ${result.get()}")
             }
         }
         .join()
```

- (a) - *Fuel* is an HTTP framework used in the examples. Important is the verb after *Fuel* - it is the REST API method.
- (b) - Adds a request header `Connection: close`, which forces a connection close after request. In the examples, this header is defined on *FuelManager* for all calls.
- (c) - Basic authentication is used for the call with user and password.

- (d) - HTTP requests with parameters require mostly JSON. The `jsonBody`-method adds the header `Content-Type: application/json` and serializes the provided object credentials to JSON. Requests without body like *DELETE* or *GET* omit body methods.
- (e) - The `responseObject<ClassType>`-method adds the header `Accept: application/json` and de-serializes the response to the specified class type. Some APIs do not have any response or non-JSON response. In such cases, the method `response` is used instead of `responseObject<ClassType>`.
- (f) - The way, how Kotlin decides between success (2xx) and failed responses. For details on the possible response status, see Configuration REST APIs [page 355].

Back to Concept [page 424]

How To Use the Examples

Some common details, used by different examples, were extracted to the `scenario.json` configuration file. This lets you use meaningful names like `config.master!!.user` in the examples. The file is loaded by

```
val config = loadScenarioConfiguration()
```

Each example also invokes

```
disableTrustChecks()
```

This method disables all SSL-related checks.

> ⚠ Caution
>
> `disableTrustChecks()` is only used for test purposes. *Do not* use it in a productive environment.

Both methods, as well as some common REST API parameter structures (data classes) are defined in the file Scenario Configuration [page 428]. This is the only help class under `sources/`.

When using the examples, start with Initial Configuration [page 429].

Prerequisite is a freshly installed Cloud Connector.

After a mandatory password change and defining the high availability role of the Cloud Connector instance (master or shadow), the example demonstrates how to provide a description for the instance and how to set up the UI and system certificates.

Once the initial configuration is done, you can optionally proceed with these steps:

- Connect to your subaccount on BTP (Subaccount Configuration [page 431])
- Create or restore a configuration backup (Backup And Restore Configuration [page 438])
- Configure and connect high availability instances (High Availability Settings [page 436])
- Integrate the Cloud Connector with the solution management infrastructure (Solution Management Integration [page 439])

Back to Concept [page 424]

Related Information

scenario.json [page 427]
Source Files [page 427]

1.4.5.2.5.13.1 scenario.json

Sample Code

```
{
    "subaccount": {
        "regionHost": "cf.eu10.hana.ondemand.com",
        "subaccount": "11aabbcc-7821-448b-9ecf-a7d986effa7c",
        "user": "xxx",
        "password": "xxx"
    },
    "master": {
        "url": "https://localhost:8443",
        "user": "Administrator",
        "password": "test"
    },
    "shadow": {
        "url": "https://localhost:8444",
        "user": "Administrator",
        "password": "test"
    }
}
```

1.4.5.2.5.13.2 Source Files

Scenario Configuration [page 428]

Initial Configuration [page 429]

Subaccount Configuration [page 431]

High Availability Settings [page 436]

Backup And Restore Configuration [page 438]

Solution Management Integration [page 439]

1.4.5.2.5.13.2.1 Scenario Configuration

Sample Code

```
package com.sap.scc.examples
import com.github.kittinunf.fuel.core.FuelError
import com.google.gson.Gson
import java.io.File
import java.security.SecureRandom
import java.security.cert.X509Certificate
import javax.net.ssl.*
data class CloudConnector(
    val url: String,
    var user: String,
    var password: String
)
fun disableTrustChecks() {
    try {
        HttpsURLConnection.setDefaultHostnameVerifier { hostname: String,
session: SSLSession -> true }
        val context: SSLContext = SSLContext.getInstance("TLS")
        val trustAll: X509TrustManager = object : X509TrustManager {
            override fun checkClientTrusted(chain: Array<X509Certificate>,
authType: String) {}
            override fun checkServerTrusted(chain: Array<X509Certificate>,
authType: String) {}
            override fun getAcceptedIssuers(): Array<X509Certificate> {
                return arrayOf()
            }
        }
        context.init(null, arrayOf(trustAll), SecureRandom())
        HttpsURLConnection.setDefaultSSLSocketFactory(context.socketFactory)
    } catch (e: Exception) {
        e.printStackTrace()
    }
}
class ScenarioConfiguration {
    var subaccount: SubaccountParameters? = null
    var master: CloudConnector? = null
    var shadow: CloudConnector? = null
}
data class SubaccountParameters(
    val regionHost: String,
    val subaccount: String,
    val user: String,
    val password: String,
    var locationId: String? = null
)
data class SccCertificate(
    var subjectDN: String? = null,
    var issuer: String? = null,
    var notAfter: String? = null,
    var notBefore: String? = null,
    var subjectAltNames: List<SubjectAltName>? = null
)
data class SubjectAltName(
    var type: String? = null,
    var value: String? = null
)
internal fun loadScenarioConfiguration(): ScenarioConfiguration {
    println("scenario.json will be loaded from $
{File("scenario.json").absolutePath}")
    return Gson().fromJson(File("scenario.json").readText(),
ScenarioConfiguration::class.java)
}
```

```
internal fun processRequestError(error: FuelError) {
    println("failed with ${error.message} ${String(error.errorData)}")
    throw RuntimeException("Stop here. ")
}
```

1.4.5.2.5.13.2.2 Initial Configuration

Sample Code

```
package com.sap.scc.examples
//import com.sap.scc.examples.SccCertificate
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.BlobDataPart
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.Method
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
import java.io.ByteArrayInputStream
import java.io.File
/*
 This example shows how to use REST APIs to perform the initial configuration
of a master instance
 after installing and starting the Cloud Connector.
 As a prerequisite you need to install and start the Cloud Connector.
 The example begins with changing the initial password, setting the instance
to the master role, edit the description,
 and upload UI and system certificates.
 For the certificates used by Cloud Connector in order to access the UI and
for the system certificate used to access backend systems,
 we simply upload the already available PKCS#12 certificates uiCert.p12 and
systemCert.p12 encrypted with the password "test1234".
 Cloud Connector also provides other options for certificate management,
please take a look at the documentation.
 The configuration details for master and shadow instances can be found in
scenario.json.
 */
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
certificate.
    //So for this demonstration use case we need to deactivate all trust
checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
    //Load configuration from property file
    val config = loadScenarioConfiguration()
    //Change the initial password
    Fuel.put("${config.master!!.url}/api/v1/configuration/connector/
authentication/basic")
        .authentication().basic(config.master!!.user, "manage")
        .body("""{"oldPassword":"manage", "newPassword":"$
{config.master!!.password}"}""")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
```

```
                is Result.Success -> println("Password successfully set to $
{config.master!!.password}")
            }
        }
        .join()
    //Set the High-Availability Role to master, use "shadow" if you want to
set it to shadow
    Fuel.put("${config.master!!.url}/api/v1/configuration/connector/haRole")
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .body("master")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("high-availability role
successfully set to 'master'")
            }
        }
        .join()
    //Edit Common Description
    Fuel.put("${config.master!!.url}/api/v1/configuration/connector")
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .body("""{"description":"<description of Cloud Connector
instance>"}""")
        .responseObject<SccDescriptionResponse> { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println(result.get())
            }
        }
        .join()
    //Upload a PKCS#12 Certificate as UI Certificate
    val uiCertificateFormData = listOf("password" to "test1234",
"keyPassword" to "test1234")
    Fuel.upload(
        "${config.master!!.url}/api/v1/configuration/connector/ui/
uiCertificate",
        Method.PUT,
        uiCertificateFormData
    )
        .add(BlobDataPart(ByteArrayInputStream(File("uiCert.p12").readBytes())
, name = "pkcs12"))
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("PKCS#12 Certificate
'uiCert.p12'  successfully uploaded")
            }
        }
        .join()
    //Upload a PKCS#12 Certificate as as System Certificate
    val systemCertificateFormData = listOf("password" to "test1234",
"keyPassword" to "test1234")
    Fuel.upload(
        "${config.master!!.url}/api/v1/configuration/connector/onPremise/
systemCertificate",
        Method.PUT,
        systemCertificateFormData
    )
        .add(BlobDataPart(ByteArrayInputStream(File("systemCert.p12").readByte
s()), name = "pkcs12"))
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .response { _, _, result ->
            when (result) {
```

```
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("PKCS#12 Certificate
'systemCert.p12'  successfully uploaded")
            }
        }
        .join()
}
//Data structures used by REST calls in this scenario
data class SccDescriptionResponse(
    var role: String,
    var description: String
)
```

1.4.5.2.5.13.2.3 Subaccount Configuration

Sample Code

```
package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.jsonBody
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
/*
 This example shows how to use REST APIs to configure and connect a
subaccount in cloud connector.
 The example begins with (1) connecting of the subaccount, then we create a
system (2) for an HTTP service
 and (3) for an RFC service.

 The configuration details for master and shadow instances can be found in
scenario.json.
 */
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
certificate.
    //So for this demonstration use case we need to deactivate all trust
checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
    //1.1. Create and connect subaccount
    //Load configuration from property file
    val config = loadScenarioConfiguration()
    //Some cloud regions require 2-Factor-Authentication
    println("Enter MFA (aka 2FA) token, if required: ")
    var token = readLine() ?: ""
    //Parameters required to establish the connection to the subaccount (aka
the secure tunnel)
    var subaccountCreateData = SubaccountConfiguration(
        config.subaccount!!.regionHost, config.subaccount!!.subaccount,
        config.subaccount!!.user, "" + config.subaccount!!.password + token,
locationId = config.subaccount!!.locationId
    )
    //Optional: Initialize the map to work with generated _links
```

```
    //If you don't like to use _links, you can easily compute the entity links
    var subaccountLinks: Map<String, HalLink> = mapOf()
    Fuel.post("${config.master!!.url}/api/v1/configuration/subaccounts")
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .jsonBody(subaccountCreateData)
        .responseObject<SubaccountInfo> { _, response, result ->
            when (result) {
                is Result.Success -> {
                    val subaccountLocation =
response.header("Location").first()
                    println("1.1. subaccount was created under:
$subaccountLocation")
                    println("subaccount: ${result.get()} ")
                    //GET details as SubaccountInfo every time possible by
invoking
                    //https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>
                    subaccountLinks = result.get()._links
                }
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
    //1.2. From time to time it is necessary to extend the validity of the
subaccount - refresh subaccount
    //Again some cloud landscapes require 2-Factor-Authentication
    println("Enter MFA (aka 2FA) token for refresh, if required: ")
    token = readLine() ?: ""
    Fuel.post(subaccountLinks["validity"]!!.href)
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .jsonBody(SubaccountRefreshCred(config.subaccount!!.user, "" +
config.subaccount!!.password + token))
        .responseObject<SubaccountInfo> { _, _, result ->
            when (result) {
                is Result.Success -> println("1.2. validity of the subaccount
was extended")
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
    //2.1. Create a system mapping (aka access control) for an HTTP service
    val httpSystem = SystemMapping(
        "virtual.host", "vport", "local", "lport",
CommunicationProtocol.HTTPS,
        BackendType.abapSys, hostInHeader = HostInHeader.INTERNAL
    )
    var httpSystemLinks: Map<String, HalLink> = mapOf()
    Fuel.post(subaccountLinks["systemMappings"]!!.href)
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .jsonBody(httpSystem)
        .responseString { _, response, result ->
            when (result) {
                is Result.Success -> {
                    val httpSystemMappingLocation =
response.header("Location").first()
                    println("2.1. system mapping was created under:
$httpSystemMappingLocation")
                    //GET the details about the system mapping by invoking
                    //https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>
                    Fuel.get(httpSystemMappingLocation)
                        .authentication().basic(config.master!!.user,
config.master!!.password)
                        .responseObject<SystemMapping> { _, _, result ->
                            when (result) {
```

```
                                is Result.Success -> {
                                    println("system mapping: ${result.get()}")
                                    httpSystemLinks = result.get()._links
                                }
                                is Result.Failure ->
processRequestError(result.error)
                            }
                        }
                        .join()
                }
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
    //2.2 Add an allowed resource to the system mapping. Since this system
mapping points to an HTTP service, use HTTP resource parameters
    Fuel.post(httpSystemLinks["resources"]!!.href)
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .jsonBody(HttpResource("/", exactMatchOnly = false))
        .responseString { _, response, result ->
            when (result) {
                is Result.Success -> {
                    val httpResourceLocation =
response.header("Location").first()
                    println("2.2. http resource was created under:
$httpResourceLocation")
                    //GET the details about the resource by invoking
                    //https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>/<encoded-id>
                    //<encoded-id> -> replace '/' with '-'
                    Fuel.get(httpResourceLocation)
                        .authentication().basic(config.master!!.user,
config.master!!.password)
                        .responseObject<HttpResource> { _, _, result ->
                            when (result) {
                                is Result.Success -> println("http resource: $
{result.get()}")
                                is Result.Failure ->
processRequestError(result.error)
                            }
                        }
                        .join()
                }
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
    //3.1 Create a system mapping for an RFC service
    var rfcSystemLinks: Map<String, HalLink> = mapOf()
    Fuel.post(subaccountLinks["systemMappings"]!!.href)
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .jsonBody(
            SystemMapping(
                "virtual.host",
                "rfcport",
                "local",
                "rfcport",
                CommunicationProtocol.RFCS,
                BackendType.abapSys
            )
        )
        .responseString { _, response, result ->
            when (result) {
                is Result.Success -> {
                    val rfcSystemMappingLocation =
response.header("Location").first()
```

```
                    println("3.1. system mapping was created under:
$rfcSystemMappingLocation")
                    //GET the details about the system mapping by invoking
                    //https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>
                    Fuel.get(rfcSystemMappingLocation)
                        .authentication().basic(config.master!!.user,
config.master!!.password)
                        .responseObject<SystemMapping> { _, _, result ->
                            when (result) {
                                is Result.Success -> {
                                    println("system mapping: ${result.get()}")
                                    rfcSystemLinks = result.get()._links
                                }
                                is Result.Failure ->
processRequestError(result.error)
                            }
                        }
                        .join()
                }
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
    //3.2 Now add the function module RFC_SYSTEM_INFO as an allowed resource
to the system mapping
    val rfcResource = RfcResource("RFC_SYSTEM_INFO")
    Fuel.post(rfcSystemLinks["resources"]!!.href)
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .jsonBody(rfcResource)
        .responseString { _, response, result ->
            when (result) {
                is Result.Success -> {
                    val resourceLocation = response.header("Location").first()
                    println("3.2. rfc resource was created under:
$resourceLocation")
                    //GET the details about the resource by invoking
                    //https://<SCC>/api/v1/configuration/subaccounts/
<regionHost>/<subaccountId>/systemMappings/<vhost>:<vport>/<encoded-id>
                    //<encoded-id> -> replace '/' with '-'
                    Fuel.get(resourceLocation)
                        .authentication().basic(config.master!!.user,
config.master!!.password)
                        .responseObject<RfcResource> { _, _, result ->
                            when (result) {
                                is Result.Success -> println("rfc resource: $
{result.get()}")
                                is Result.Failure ->
processRequestError(result.error)
                            }
                        }
                        .join()
                }
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
}
//Data structures used by REST calls in this scenario
//Nullable properties are optional
data class SubaccountConfiguration(
    var regionHost: String,
    var subaccount: String,
    var cloudUser: String,
    var cloudPassword: String,
    var displayName: String? = null,
    var locationId: String? = null
```

```
)
data class SubaccountInfo(
    val displayName: String,
    val regionHost: String,
    val subaccount: String,
    val tunnel: SubaccountTunnelInfo,
    val user: String,
    val _links: Map<String, HalLink>
)
data class SubaccountTunnelInfo(
    val state: String, //Connected
    val connectedSince: String, //timestamp formatted with client locale
    val connections: Int,
    val applicationConnections: List<String>,
    val serviceChannels: List<String>,
    val subaccountCertificate: X509CertificateInfo,
    )
data class X509CertificateInfo(
    val notAfter: String,
    val notBefore: String,
    val subjectDN: String,
    val issuer: String
)
data class HalLink(val href: String)
data class SubaccountRefreshCred(
    var cloudUser: String,
    var cloudPassword: String
)
data class SystemMapping(
    val virtualHost: String,
    val virtualPort: String,
    val localHost: String,
    val localPort: String,
    val protocol: CommunicationProtocol,
    val backendType: BackendType,
    val sncPartnerName: String? = null,
    val hostInHeader: HostInHeader? = null,
    val sapRouter: String? = null,
    val authenticationMode: AuthenticationMode = AuthenticationMode.NONE,
    val description: String = "",
) {
    val _links: Map<String, HalLink> = mapOf()
}
enum class CommunicationProtocol {
    HTTP, HTTPS, RFC, RFCS, LDAP, LDAPS, TCP, TCPS
}
enum class BackendType {
    abapSys, netweaverCE, applServerJava, BC, PI, hana, netweaverGW,
otherSAPsys, nonSAPsys
}
enum class HostInHeader {
    INTERNAL, VIRTUAL
}
enum class AuthenticationMode {
    NONE, X509_CERTIFICATE, X509_CERTIFICATE_LOCAL, KERBEROS
}
data class HttpResource(
    val id: String, //URI path
    val enabled: Boolean = true,
    val exactMatchOnly: Boolean = true,
    val webSocketUpgradeAllowed: Boolean = true,
    val description: String = ""
)
data class RfcResource(
    val id: String, //Function module or prefix for function module
    val enabled: Boolean = true,
    val exactMatchOnly: Boolean = true,
    val description: String = ""
```

```
)
```

1.4.5.2.5.13.2.4 High Availability Settings

Sample Code

```
package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.Headers
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.jsonBody
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
import java.net.URL
/*
 This example shows how to use REST APIs to change high availability settings
of the Cloud Connector instances.
 As a prerequisite you need to install and start a shadow and a master
instance.
 Afterwards perform the initial configuration of the master instance (see
InitialConfiguration.kt).

 The configuration details for master and shadow instances can be found in
scenario.json.
 */
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
certificate.
    //So for this demonstration use case we need to deactivate all trust
checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
    //Load configuration from property file
    val config = loadScenarioConfiguration()
    //The high-availability role 'master' in Cloud Connector instance
'master' is already set in the initial configuration.
    //Set the high-availability role to 'shadow' in Cloud Connector instance
'shadow'
    Fuel.put("${config.shadow!!.url}/api/v1/configuration/connector/haRole")
        .authentication().basic(config.shadow!!.user,
config.shadow!!.password)
        .body("shadow")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("high-availability role
successfully set to 'shadow'")
            }
        }
        .join()
    //Enable HA in the master instance and define the allowed shadow hosts
    Fuel.put("${config.master!!.url}/api/v1/configuration/connector/ha/master/
config")
        .header(Headers.CONTENT_TYPE, "application/json")
```

```
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .body("""{"haEnabled":"true", "allowedShadowHost":"$
{URL(config.shadow!!.url).host}"}""")
        .responseObject<HAMasterConfiguration> { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> {
                    val masterConfig = result.get()
                    println("Master configuration: $masterConfig ")
                }
            }
        }
        .join()
    //The configuration of the shadow instance
    var shadowConfig: HAShadowConfiguration? = null
    //Get the current configuration
    Fuel.get("${config.shadow!!.url}/api/v1/configuration/connector/ha/shadow/
config")
        .authentication().basic(config.shadow!!.user,
config.shadow!!.password)
        .responseObject<HAShadowConfiguration> { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> shadowConfig = result.get()
            }
        }
        .join()
    //Edit the retrieved configuration and set it
    shadowConfig!!.masterPort = "${URL(config.master!!.url).port}"
    shadowConfig!!.masterHost = URL(config.master!!.url).host
    shadowConfig!!.ownHost = URL(config.master!!.url).host
    Fuel.put("${config.shadow!!.url}/api/v1/configuration/connector/ha/shadow/
config")
        .authentication().basic(config.shadow!!.user,
config.shadow!!.password)
        .jsonBody(shadowConfig!!)
        .responseObject<HAShadowConfiguration> { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> shadowConfig = result.get()
            }
        }
        .join()
    //Set the HA link of both cloud connector instances to 'CONNECT'
    Fuel.post("${config.shadow!!.url}/api/v1/configuration/connector/ha/
shadow/state")
        .header(Headers.CONTENT_TYPE, "application/json")
        .authentication().basic(config.shadow!!.user,
config.shadow!!.password)
        .body("""{"op":"CONNECT", "user":"${config.master!!.user}",
"password":"${config.master!!.password}"}""")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("Master (${config.master!!.url})
and shadow (${config.shadow!!.url}) successfully connected")
            }
        }
        .join()
}
//Data structures used by REST calls in this scenario
//Structure used  to read the high availability settings for a Cloud
Connector master instance
data class HAMasterConfiguration(
    var haEnabled: Boolean? = null,
    var allowedShadowHost: String? = null
)
```

```
//Structure used  to read and edit the high availability settings for a Cloud
Connector shadow instance
data class HAShadowConfiguration(
    var masterPort: String,
    var masterHost: String,
    var ownHost: String? = null,
    var checkIntervalInSeconds: Int?,
    var takeoverDelayInSeconds: Int?,
    var connectTimeoutInMillis: Int?,
    var requestTimeoutInMillis: Int?
)
```

1.4.5.2.5.13.2.5 Backup And Restore Configuration

Sample Code

```
package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.BlobDataPart
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.Headers
import com.github.kittinunf.fuel.core.Method
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.result.Result
import java.io.ByteArrayInputStream
import java.io.File
/*
 This example shows how to use REST APIs to perform the Backup and Restore of
the Cloud Connector configuration.
 As a prerequisite you have a stable Cloud Connector configuration and you
want to save the backup for later use.

 The configuration details can be found in scenario.json.
 */
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
certificate.
    //So for this demonstration use case we need to deactivate all trust
checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
    //Load configuration from property file
    val config = loadScenarioConfiguration()
    //Create the backup configuration of the cloud connector "master".
    Fuel.post("${config.master!!.url}/api/v1/configuration/backup")
        .header(Headers.CONTENT_TYPE, "application/json")
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .body("""{"password":"my-very-secret-password"}""")
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success ->
File("backup.zip").writeBytes(result.get())
            }
```

```
        }
        .join()
    //Restore the backup configuration if some fatal accidental changes
should be reverted
    val formData = listOf("password" to "my-very-secret-password")
    Fuel.upload("${config.master!!.url}/api/v1/configuration/backup",
Method.PUT, formData)
        .add(BlobDataPart(ByteArrayInputStream(File("backup.zip").readBytes())
, name = "backup"))
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .response { _, _, result ->
            when (result) {
                is Result.Failure -> processRequestError(result.error)
                is Result.Success -> println("Backup configuration
successfully restored in (${config.master!!.url}) ")
            }
        }
        .join()
}
```

1.4.5.2.5.13.2.6 Solution Management Integration

Sample Code

```
package com.sap.scc.examples
import com.github.kittinunf.fuel.Fuel
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.core.extensions.authentication
import com.github.kittinunf.fuel.gson.jsonBody
import com.github.kittinunf.fuel.gson.responseObject
import com.github.kittinunf.result.Result
import java.io.File
/*
 This example shows how to use REST APIs to configure the integration with
SAP solution management.
 In most cases it should be only necessary to pass a boolean flag in order to
enable or
 disable solution management integration. It is expected that SAP host agent
is already
 installed on the host as prerequisite for solution management integration.

 This example executes requests against master and shadow instances.
 The shadow instance is optional. Ignore the requests to shadow instance if
there is only
 a master instance in your environment.

 The configuration details for master and shadow instances can be found in
scenario.json.
 */
fun main() {
    //Cloud Connector distribution generates only an untrusted self-signed
certificate.
    //So for this demonstration use case we need to deactivate all trust
checks.
    disableTrustChecks()
    //Use 'Connection: close' header, to make stateless communication more
efficient
    FuelManager.instance.baseHeaders = mapOf("Connection" to "close")
```

```
    //Add output of cURL commands for revision
    //FuelManager.instance.addRequestInterceptor(LogRequestAsCurlInterceptor)
    //Load configuration from property file
    val config = loadScenarioConfiguration()
    //First we check the current configuration of solution management
    Fuel.get("${config.master!!.url}/api/v1/configuration/connector/
solutionManagement")
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .responseObject<SolutionManagementConfiguration> { _, _, result ->
            when (result) {
                is Result.Success -> println("response: ${result.get()}")
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
    //Configure the hostAgent path on the shadow instance
    //Invoking this API on a shadow instance changes only the configuration.
    //Only when invoking it on a master instance it also activates solution
management integration.
    //Note: This step is not required if host agent is installed at the
default location
    Fuel.post("${config.shadow!!.url}/api/v1/configuration/connector/
solutionManagement")
        .authentication().basic(config.shadow!!.user,
config.shadow!!.password)
        .jsonBody(SolutionManagementConfiguration(hostAgentPath = "/path/to/
hostAgent"))
        .response { _, _, result ->
            when (result) {
                is Result.Success -> println("new path to host agent was set")
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
    //Turns on the solution management integration.
    //Note: this operation is possible only on the master instance.
    //In case of an HA setup, the flag enabled is propagated to the shadow
automatically.
    //Note: optionally you can set here the new path for host agent and
enable DSR.
    Fuel.post("${config.master!!.url}/api/v1/configuration/connector/
solutionManagement")
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .jsonBody(SolutionManagementConfiguration())
        .response { _, _, result ->
            when (result) {
                is Result.Success -> println("solution management is
activated")
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
    //Turns off the solution management integration.
    //Note: this operation is possible only on the master instance.
    //In case of an HA setup, the flag not enabled is propagated to the
shadow automatically
    Fuel.delete("${config.master!!.url}/api/v1/configuration/connector/
solutionManagement")
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .response { _, _, result ->
            when (result) {
                is Result.Success -> println("solution management is
deactivated")
                is Result.Failure -> processRequestError(result.error)
            }
```

```
        }
        .join()
    //Registration file with LMDB model can be downloaded by following
request.
    //This file can be used for a manual upload to solution management.
    //If solution management integration is active,the updates are triggered
by configuration changes automatically.
    Fuel.get("${config.master!!.url}/api/v1/configuration/connector/
solutionManagement/registrationFile")
        .authentication().basic(config.master!!.user,
config.master!!.password)
        .response { _, _, result ->
            when (result) {
                is Result.Success ->
File("lmdbModel.zip").writeBytes(result.get())
                is Result.Failure -> processRequestError(result.error)
            }
        }
        .join()
}
//Data structures used by REST calls in this scenario
//Nullable properties are optional
data class SolutionManagementConfiguration(
    var hostAgentPath: String? = null,
    var enabled: Boolean? = null,
    var dsrEnabled: Boolean? = null
)
```

1.4.5.2.6 Configure an On-Premise User Store

Configure SAP BTP Java applications to use your corporate LDAP server or on-premise SAP system as a user store.

Prerequisites

- Your have configured your Java cloud application to use an on-premise user provider and to consume its users via the Cloud Connector. To do this, execute the following command:

  ```
  neo deploy --host <host> --account <subaccount name> --application
  <application name> --source <path to WAR file> --user <e-mail or user name> --
  vm-arguments "-Dcom.sap.cloud.security.um.user_provider_name=onpremise -
  Dcom.sap.cloud.security.um.destination_name=onpremiseumconnector"
  ```

- You have created a connectivity destination to configure the on-premise user provider, using the following paremeters:

  ```
  Name=onpremiseumconnector
  Type=HTTP
  URL= http://scc.scim:80/scim/v1
  Authentication=NoAuthentication
  CloudConnectorVersion=2
  ProxyType=OnPremise
  ```

- You are using only one domain for user authentication. Authentication to multiple domains including sub-domains is not supported.

For more information, see also On-Premise User Store [page 1736].

Context

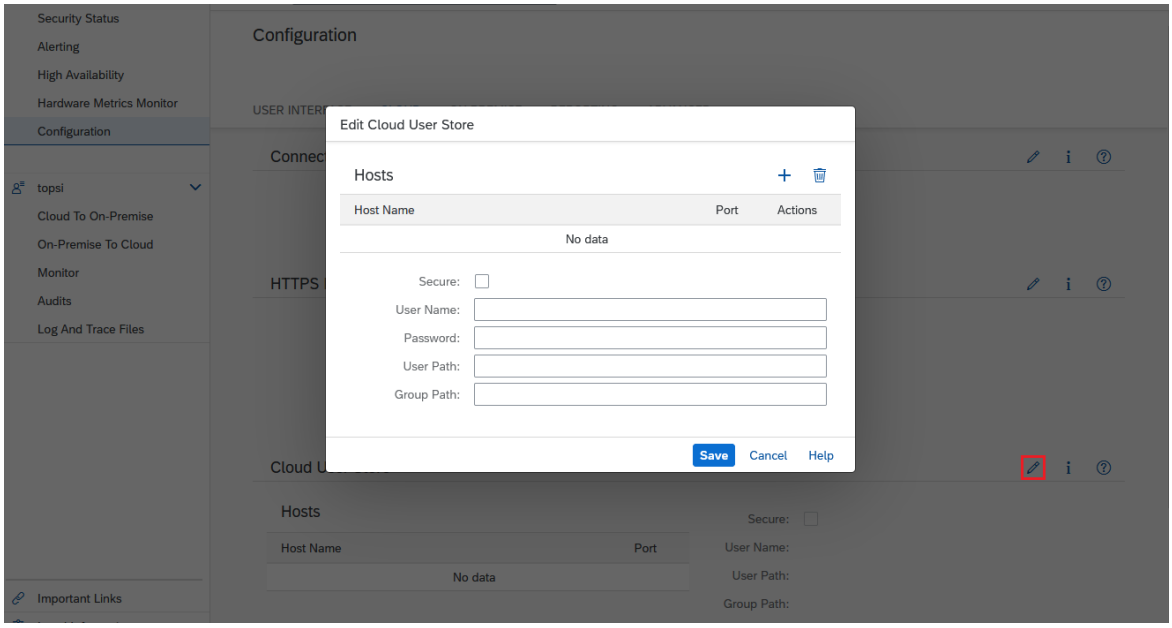
If you configure your SAP BTP applications to use the corporate LDAP server or on-premise SAP system as a user store, the platform doesn't need to keep the entire user database but requests the necessary information from the on-premise user store. Java applications running on SAP BTP can use the on-premise system to check credentials, search for users, and retrieve details. In addition to the user information, the cloud application may request information about the groups a user belongs to.

One way a Java cloud application can define user authorizations is by checking a user's membership to specific groups in the on-premise user store. The application uses the roles for the groups defined in SAP BTP. For more information, see Managing Roles [page 1710].

> **i Note**
>
> The configuration steps below are applicable only for Microsoft Active Directory (AD).

Procedure

1. From the main menu, choose *Configuration*.
2. From the *Cloud User Store* section on the *Cloud* tab, choose *Edit*.



3. To connect to LDAP using SSL, select *Secure*.
4. Manage the hosts and ports for your LDAP servers:

- Choose the *Add* icon to add a host.

  > **Note**
  >
  > Multiple hosts are currently not supported.

- Choose *Edit* to edit the selected host.
- Choose *Delete* to delete the selected hosts.

5. Enter the user name and password for the service user that is used to contact the LDAP system.

   > **Note**
   >
   > The user name must be fully qualified, including the AD domain suffix, for example, `john.smith@mycompany.com`.

6. In *User Path*, specify the LDAP subtree that contains the users.
7. In *Group Path*, specify the LDAP subtree that contains the groups.
8. Choose **Save**.

Related Information

Using an SAP System as an On-Premise User Store [page 1737]
Use LDAP for Authentication [page 466]
Managing Destinations

1.4.5.2.7 Using Service Channels

Configure Cloud Connector service channels to connect your on-premise network to specific services on SAP BTP or to S/4HANA Cloud.

Context

Cloud Connector service channels provide access from an external network to certain services on SAP BTP, or to S/4HANA Cloud. The called services are not exposed to direct access from the Internet. The Cloud Connector ensures that the connection is always available and communication is secured.

Service Channel Type	Description
SAP HANA Database on SAP BTP	The service channel for the **SAP HANA Database** lets you access SAP HANA databases that run in the Cloud from database clients (for example, clients using ODBC/JDBC drivers). You can use the service channel to connect database, analytical, BI, or replication tools to your SAP HANA database in your SAP BTP subaccount.
Virtual Machine on SAP BTP	You can use the **virtual machine** (VM) service channel to access a SAP BTP VM using an SSH client, and adjust it to your needs.
RFC Connection to SAP BTP ABAP environment and S/4HANA Cloud	The service channel for **RFC** supports calls from on-premise systems to the SAP BTP ABAP environment and S/4HANA Cloud using RFC.

Next Steps

Configure a Service Channel for an SAP HANA Database [page 444]

Connect DB Tools to SAP HANA via Service Channels [page 447]

Configure a Service Channel for Virtual Machine [page 448]

Configure a Service Channel for RFC [page 450]

Related Information

Service Channels: Port Overview [page 451]

1.4.5.2.7.1 Configure a Service Channel for an SAP HANA Database

Using Cloud Connector service channels, you can establish a connection to an SAP HANA database in SAP BTP that is not directly exposed to external access.

Context

The service channel for SAP HANA Database lets you access SAP HANA databases running in the cloud via ODBC/JDBC. You can use the service channel to connect database, analytical, BI, or replication tools to an SAP HANA database in your SAP BTP subaccount.

Restrictions

This feature is only available in regions (data centers) that provide the SAP HANA service (that is, SAP data centers (Neo environment), Azure and AWS).

For more information, see SAP HANA Service for SAP BTP Getting Started Guide.

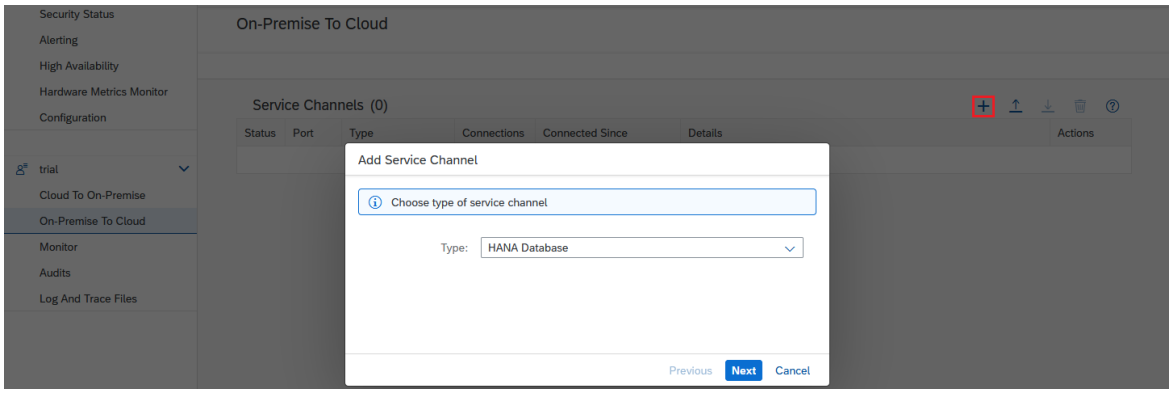
- In AWS regions, you can currently use the service channel only for the SAP HANA service provisioned *before June 4, 2018* (old version).
- If you are using the SAP HANA service provisioned *after June 4, 2018* (new version, available on AWS and GCP) or the *SAP HANA Cloud service*, you can currently access SAP HANA databases only directly, without going through the Cloud Connector.
  For more information, see Connecting to an SAP HANA Service Instance Directly from SAP HANA Clients.

To find detailed information for a specific Cloud Foundry region and SAP HANA service version, see Find the Right Guide.

> **i Note**
>
> The following procedure requires a productive SAP HANA instance that is available in the *same* subaccount. You cannot access an SAP HANA instance that is owned by a different subaccount within the same or another global account (shared SAP HANA database). See also Sharing Databases with Other Subaccounts.

Procedure

1. From your subaccount menu, choose *On-Premise To Cloud*.
2. Choose *Add* (+).



3. In the *Add Service Channel* dialog, leave the default value **`HANA Database`** in the `<Type>` field.
4. Choose *Next*.
5. Choose the SAP HANA instance name. If you cannot select it from the drop-down list, enter the instance name manually. In the **Neo** environment, it must match one of the names (IDs) shown in the cockpit under *SAP HANA/SAP ASE* > *Databases & Schemas*, in the `<DB/Schema ID>` column.

   > **i Note**
   >
   > The SAP HANA instance name is case-sensitive.

In the **Cloud Foundry** environment, the format of the SAP HANA instance name includes the Cloud Foundry space name, the database name and the database ID.

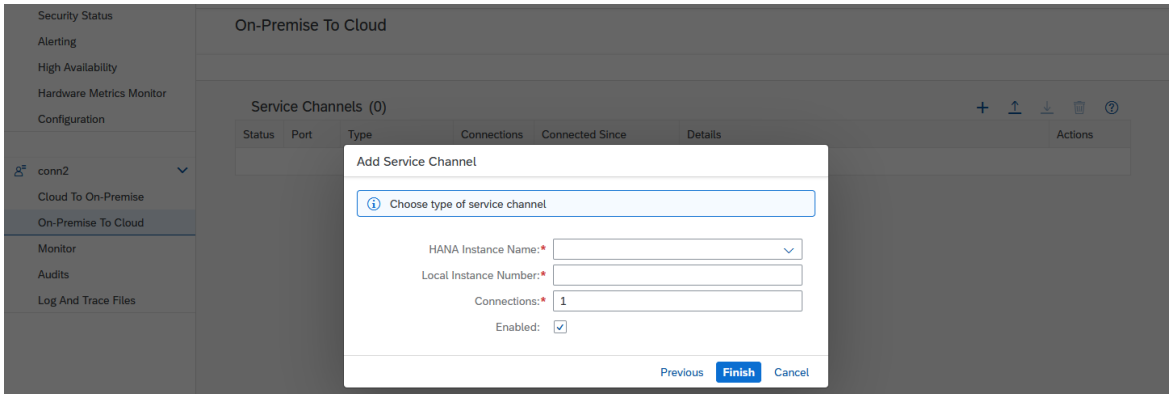
Example:

test:testInstance:3fcc976d-457a-474e-975b-e572600f474e:de19c262-a1fc-4096-bfce-1c41388e4b49

where

- *test*: Cloud Foundry space name
- *testInstance*: tenant database instance name
- *3fcc976d-457a-474e-975b-e572600f474e:de19c262-a1fc-4096-bfce-1c41388e4b49*: database ID

6. Specify the local instance number. This is a double-digit number which computes the local port used to access the SAP HANA instance in the cloud. The local port is derived from the local instance number as **`3<instance number>15`**. For example, if the instance number is **`22`**, then the local port is **`32215`**. The local instance number must not be **`00`**. This instance number might cause problems with some SAP HANA clients.
7. Specify the number of physical connections to SAP BTP. The default value is **`1`**. One physical connection can open various virtual connections through multiplexing.



8. Leave *Enabled* selected to establish the channel immediately after clicking *Finish*, or unselect it if you don't want to establish the channel immediately.
9. Choose *Finish*.

Next Steps

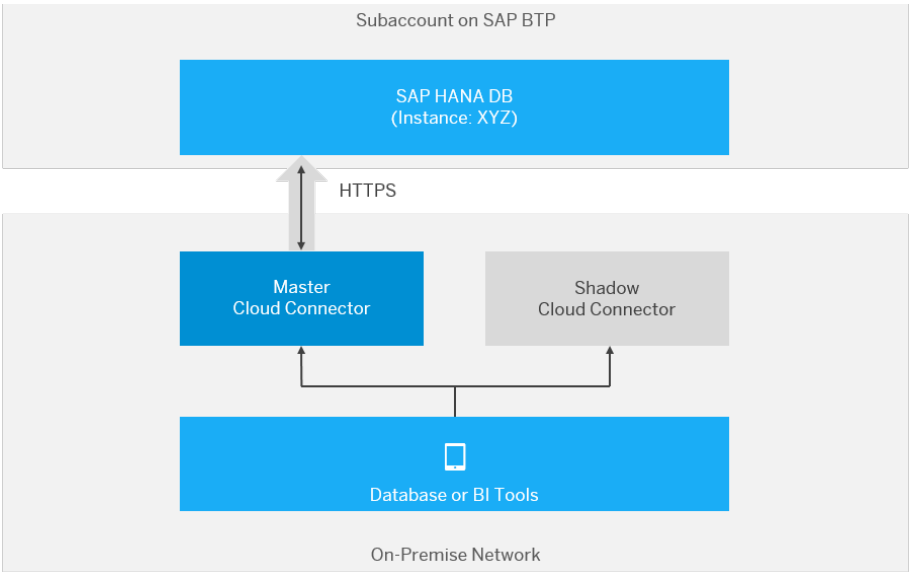
Once you have established an SAP HANA Database service channel, you can connect on-premise database or BI tools to the selected SAP HANA database in the cloud. This can be done by using **`<cloud_connector_host>:<local_HANA_port>`** in the JDBC/ODBC connect strings.

See Connect DB Tools to SAP HANA via Service Channels [page 447].

1.4.5.2.7.2 Connect DB Tools to SAP HANA via Service Channels

Context

You can connect database, BI, or replication tools running in on-premise network to an SAP HANA database on SAP BTP using service channels of the Cloud Connector. You can also use the high availability support of the Cloud Connector on a database connection. The picture below shows the landscape in such a scenario.



Follow the steps below to set up failover support, configure a service channel, and connect on-premise DB tools via JDBC or ODBC to the SAP HANA database.

- For more information on using SAP HANA instances, see Using an SAP HANA XS Database System [page 999]
- For the connection string via ODBC you need a corresponding database user and password (see step 4 below). See also: Creating Database Users [page 1003].
- Find detailed information on failover support in the *SAP HANA Administration Guide*: Configuring Clients for Failover.

> **i Note**
>
> This link points to the latest release of *SAP HANA Administration Guide*. Refer to the SAP BTP Release Notes to find out which SAP HANA SPS is supported by SAP BTP. Find the list of guides for earlier releases in the *Related Links* section below.

Procedure

1. To establish a highly available connection to one or multiple SAP HANA instances in the cloud, we recommend that you make use of the failover support of the Cloud Connector. Set up a master and a shadow instance. See Install a Failover Instance for High Availability [page 470].
2. In the master instance, configure a service channel to the SAP HANA database of the SAP BTP subaccount to which you want to connect. If, for example, the chosen HANA instance is 01, the port of the service channel is 30115. See also Configure a Service Channel for an SAP HANA Database [page 444].
3. Connect on-premise DB tools via JDBC to the SAP HANA database by using the following connection string:

   Example:

   ```
   jdbc:sap://<cloud-connector-master-host>:30115;<cloud-connector-shadow-host>:
   30115[/?<options>]
   ```

   The SAP HANA JDBC driver supports failover out of the box. All you need is to configure the shadow instance of the Cloud Connector as a failover server in the JDBC connection string. The different options supported in the JDBC connection string are described in: Connect to SAP HANA via JDBC
4. You can also connect on-premise DB tools via ODBC to the SAP HANA database. Use the following connection string:

   ```
   "DRIVER=HDBODBC32;UID=<user>;PWD=<password>;SERVERNODE=<cloud-connector-
   master-host>:30115;<cloud-connector-shadow-host>:30115;"
   ```

Related Information

Guides for earlier releases of SAP HANA

1.4.5.2.7.3 Configure a Service Channel for Virtual Machine

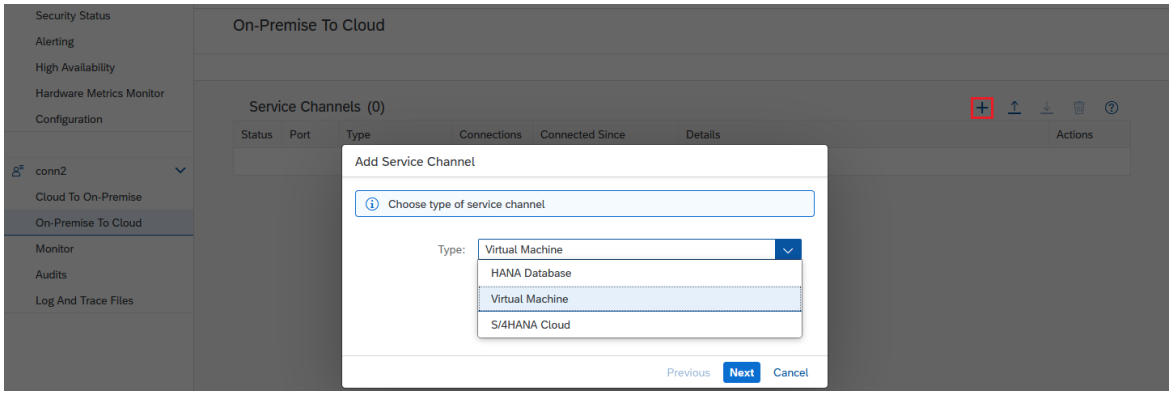
Context

You can establish a connection to a virtual machine (VM) in the SAP BTP that is not directly exposed to external access. Use *On-Premise to Cloud* > *Service Channels* in the Cloud Connector. You can use the service channel to manage the VM and adjust it to your needs.

> **i Note**
>
> The following procedure requires that you have created a VM in your subaccount.

Procedure

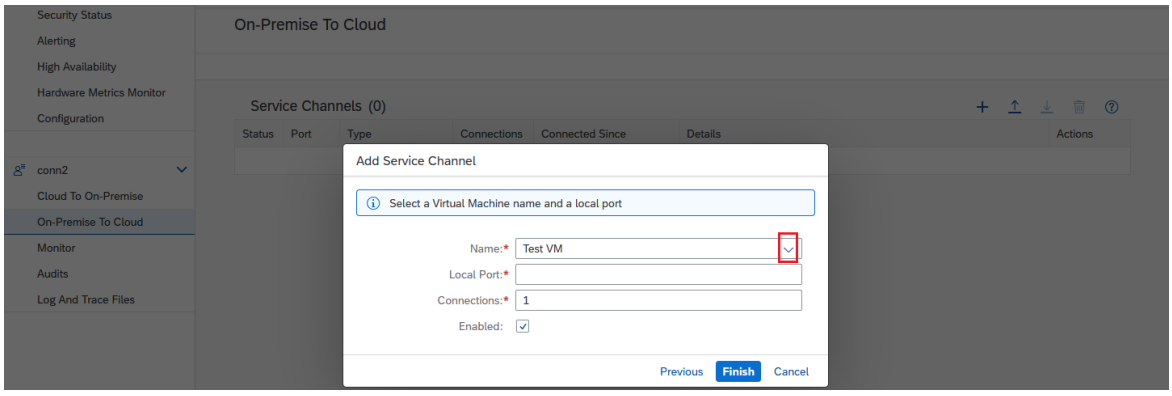
1. From your subaccount menu, choose *On Premise To Cloud*.
2. Choose the *Add* icon.



3. In the *Add Service Channel* dialog, select **`Virtual Machine`** from the list of supported channel types.
4. Choose *Next*. The *Virtual Machine* dialog opens.
5. Choose the Virtual Machine `<Name>` from the list of available Virtual Machines. It matches the corresponding name shown under *Virtual Machines* in the cockpit.

> **i Note**
>
> The Virtual Machine name is case-sensitive.

6. Choose the `<Local Port>`. You can use any port that is not used yet.



7. Leave `<Enabled>` selected to establish the channel immediately after clicking *Save*. Unselect it if you don't want to establish the channel immediately.
8. Choose *Finish*.

Next Steps

Once you have established a service channel for the Virtual Machine, you can connect it with your SSH client. This may be done by accessing `<Cloud_connector_host>:<local_VM_port>` and the key file that was generated when creating the virtual machine.

1.4.5.2.7.4 Configure a Service Channel for RFC

For scenarios that need to call from on-premise systems to SAP BTP ABAP environment or to S/4HANA Cloud using RFC, you can establish a connection to an ABAP Cloud tenant host. To do this, select *On-Premise to Cloud* > *Service Channels* in the Cloud Connector.

Prerequisites

S/4HANA Cloud

You have set up the S/4HANA Cloud environment for communication with the Cloud Connector.

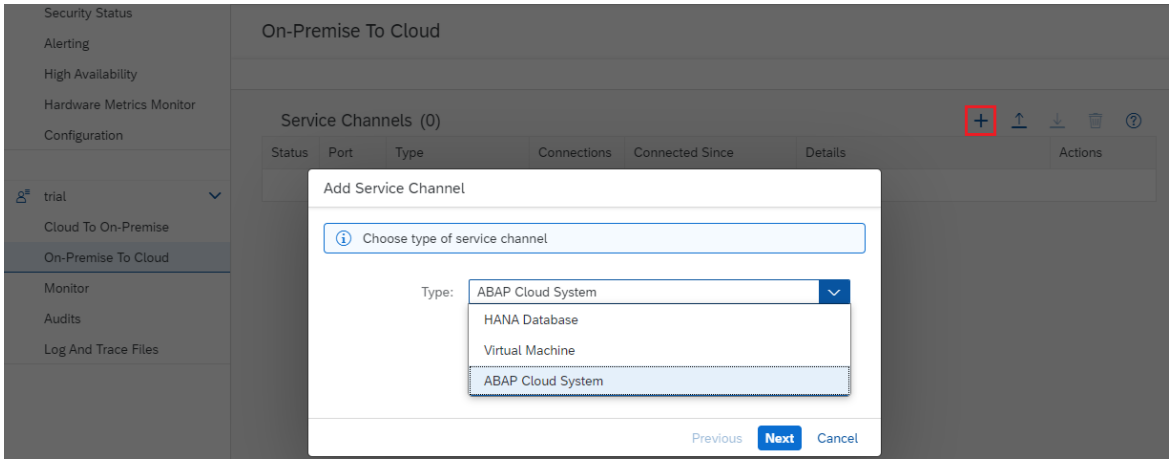
In particular, you must create a communication arrangement for the scenario SAP_COM_0200 (SAP Cloud Connector Integration). See Integrating On-Premise Systems (SAP S/4HANA Cloud documentation).

SAP BTP ABAP Environment

- When using the default connectivity setup with the **Cloud Foundry** subaccount in which the system has been provisioned, you can use a service channel without additional configuration, as long as the system is a single-tenant system.
- When using connectivity via a **Neo** subaccount, you must create, like for S/4HANA Cloud, a communication arrangement for the scenario SAP_COM_0200. For more information, see Create a Communication Arrangement for Cloud Connector Integration (documentation for ABAP environment on SAP BTP).

Procedure

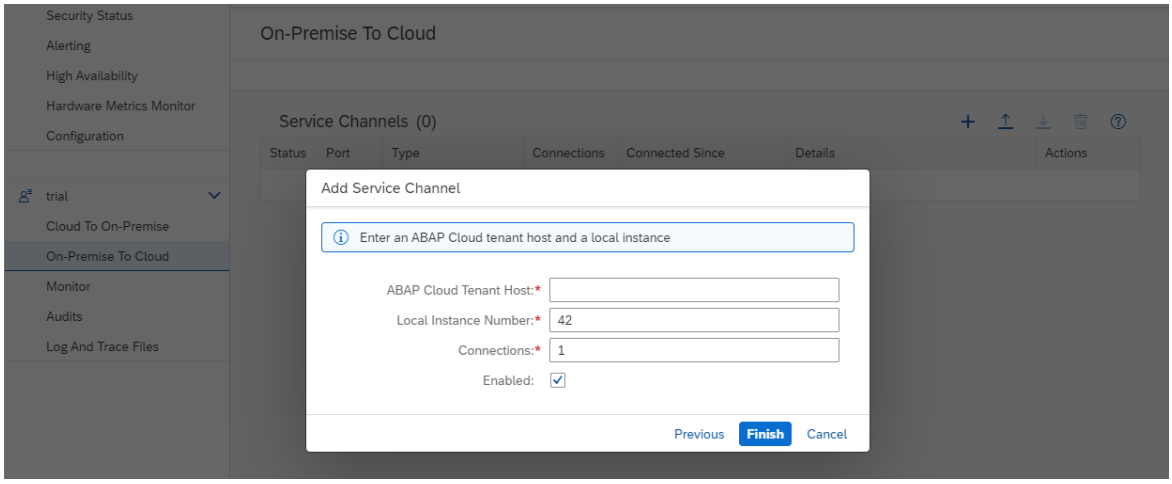
1. From your subaccount menu, choose *On Premise To Cloud*.
2. Choose the *Add* (+) icon.



3. In the *Add Service Channel* dialog, select **`ABAP Cloud System`** from the drop-down list of supported channel types.
4. Choose *Next*. The *ABAP Cloud System* dialog opens.
5. Enter the `<ABAP Cloud Cloud Tenant Host>` that you want to connect to.

> **i Note**
>
> The S/4HANA Cloud tenant host name is case-sensitive. Also make sure that you specify the API address of your tenant host. For example, if the tenant host of your instance is **`<my1234567>.s4hana.ondemand.com`**, the API tenant host to be specified is **`<my1234567>-api.s4hana.ondemand.com`**.

6. In the same dialog window, define the `<Local Instance Number>` under which the ABAP Cloud system is reachable for the client systems. You can enter any instance number for which the port is not used yet on the Cloud Connector host. The port numbers result from the following pattern: **`33<LocalInstanceNumber>`**.
7. In the same dialog window, leave *Enabled* selected to establish the channel immediately after choosing *Finish*. Unselect it if you don't want to establish the channel immediately.



8. Choose *Finish*.

> **i Note**
>
> When addressing an ABAP Cloud system in a **destination configuration**, you must enter the *Cloud Connector host* as application server host. As instance number, specify the `<Local Instance Number>` that you configured for the service channel. As user, you must provide the *business user* name but not the technical user name associated with the same.

1.4.5.2.7.5 Service Channels: Port Overview

A service channel overview lets you see the details of all service channels that are used by a Cloud Connector installation.

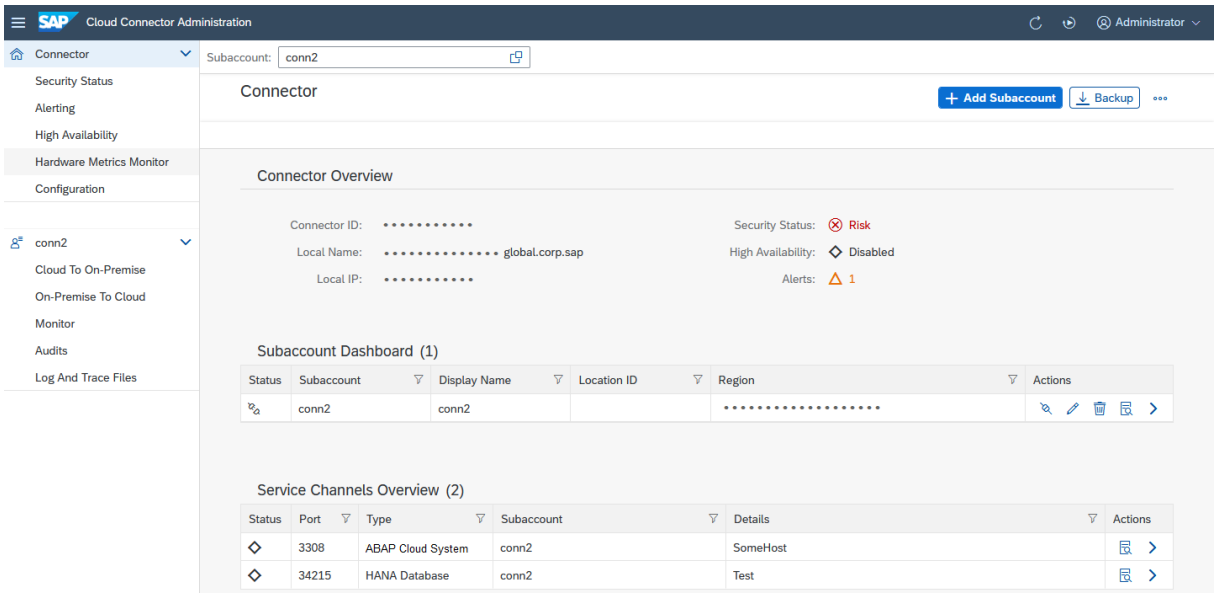
The service channel port overview lists all service channels that are configured in the Cloud Connector. It lets you see at a glance, which server ports are used by a Cloud Connector installation.

In addition, you can find the following information about each service channel:

- Status (`enabled`, `disabled`, `disconnected`)
- Service channel type (`SAP HANA Database`, `Virtual Machine`, `ABAP Cloud System`)
- Assigned subaccount
- Details (for example, the assigned SAP HANA instance name or virtual machine name)

From the *Actions* column, you can switch directly to the *On-Premise To Cloud* section of the corresponding subaccount and edit the selected service channel.

To find the overview list, choose *Connector* from the navigation menu and go to section *Service Channels Overview*:



1.4.5.2.8 Configure Trust

Set up an allowlist for cloud applications and a trust store for on-premise systems in the Cloud Connector.

Tasks

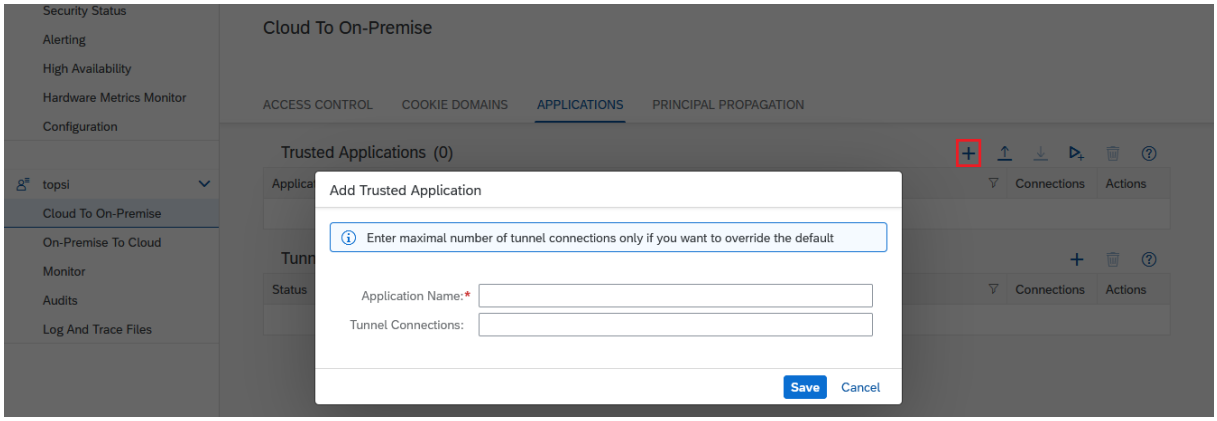
Trust Cloud Applications in the Cloud Connector [page 453]

Configure the Trust Store [page 454]

Trust Cloud Applications in the Cloud Connector

> ! Restriction
>
> Currently, the complete implementation of this feature is available only for interaction with the Neo environment.

By default, all applications within a subaccount are allowed to use the Cloud Connector associated with the subaccount they run in. However, this behavior might not be desired in any scenario. For example, this may be acceptable for some applications, as they must interact with on-premise resources, while other applications, for which it is not transparent whether they try to receive on-premise data, might turn out to be malicious. For such cases, you can use an application allowlist.



As long as there is no entry in this list, all applications are allowed to use the Cloud Connector. If one or more entries appear in the allowlist, then only these applications are allowed to connect to the exposed systems in the Cloud Connector.

You can add, edit, or delete entries as follows:

1. From your subaccount menu, choose *Cloud to On-Premise* and go to the *Applications* tab.
2. To add an application, choose the *Add* icon in section *Trusted Applications*.
3. Enter the `<Application Name>` in the *Add Tunnel Application* dialog.

   > i Note
   >
   > To add all applications that are listed in section *Tunnel Connection Limits* on the same screen, you can also use the *Upload* button next to the *Add* button. The list *Tunnel Connection Limits* shows all applications for which a specific maximal number of tunnel connections was specified. See also: Configure Tunnel Connections [page 458].

4. (Optional) Enter the maximal number of `<Tunnel Connections>` only if you want to override the default value.
5. Choose *Save*.

   > i Note
   >
   > The application name is visible in the SAP BTP cockpit under *Applications* > *Java Applications*. To allow a subscribed application, you must add it to the allowlist in the format

> **<providerSubaccount>:<applicationName>**. In particular, when using HTML5 applications, an implicit subscription to *services:dispatcher* is required.

To edit an existing entry:

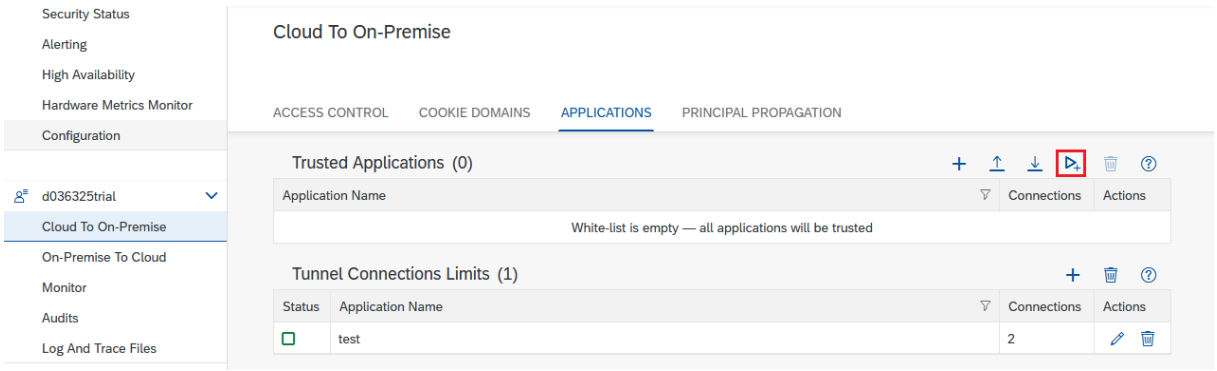
1. Choose the *Edit* button.
2. When you are done, select *Save*.

To remove an application from the list:

1. Select the entry.
2. Choose *Delete*.

To delete all entries, choose *Delete All*.

To add all applications from section *Tunnel Connection Limits* to the allowlist, choose the button *Add all applications...* from section **Trusted Applications**.



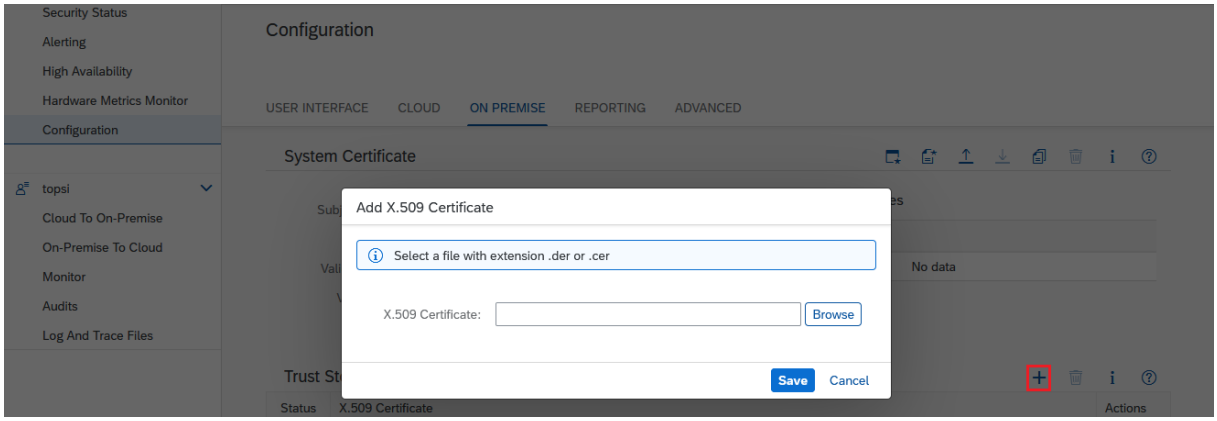
Back to Tasks [page 452]

Configure the Trust Store

By default, the Cloud Connector trusts every on-premise system when connecting to it via TLS. As this may be an undesirable behavior from a security perspective, you can configure a trust store that acts as an allowlist of trusted certificate authorities. Any TLS server certificate issued by one of those CAs will be considered trusted. If the CA that has issued a concrete server certificate is not included in the trust store, the server is considered untrusted and the connection will fail.

You can configure the trust store as follows:

1. From the main menu, choose *Configuration* .
2. Select the *On Premise* tab, section *Trust Store*.



An empty trust store does not impose any restrictions on the trusted on-premise systems. It becomes an allowlist as soon as you add the first public key.

> **i Note**
>
> You must provide the CA's X.509 certificates in `.der` or `.cer` format.

Back to Tasks [page 452]

1.4.5.2.9 Configure Domain Mappings for Cookies

Context

Some HTTP servers return cookies that contain a `domain` attribute. For subsequent requests, HTTP clients should send these cookies to machines that have host names in the specified domain.

For example, if the client receives a cookie like the following:

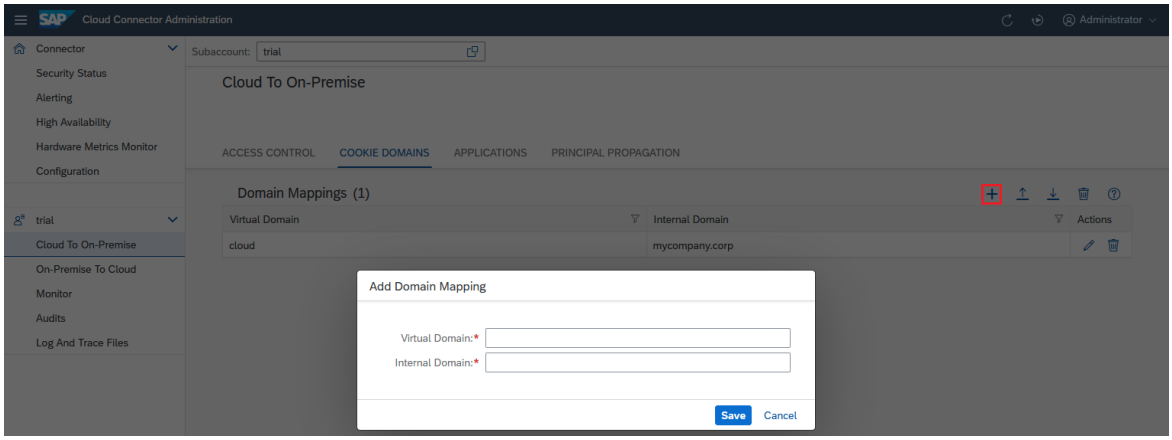
```
Set-Cookie: cookie-field=some-value; domain=mycompany.corp; path=...; ...
```

It returns the cookie in follow-up requests to all hosts like **`ecc60.mycompany.corp`**, **`crm40.mycompany.corp`**, and so on, if the other attributes like `path` and `attribute` require it.

However, in a Cloud Connector setup between a client and a Web server, this may lead to problems. For example, assume that you have defined a virtual host *sales-system.cloud* and mapped it to the internal host name **`ecc60.mycompany.corp`**. The client "thinks" it is sending an HTTP request to the host name *sales-system.cloud*, while the Web server, unaware of the above host name mapping, sets a cookie for the domain **`mycompany.corp`**. The client does not know this domain name and thus, for the next request to that Web server, doesn't attach the cookie, which it should do. The procedure below prevents this problem.

Procedure

1. From your subaccount menu, choose *Cloud To On-Premise*, and go to the *Cookie Domains* tab.
2. Choose *Add*.
3. Enter **`cloud`** as the virtual domain, and your company name as the internal domain.
4. Choose *Save*.



The Cloud Connector checks the Web server's response for **`Set-Cookie`** headers. If it finds one with an attribute `domain=intranet.corp`, it replaces it with `domain=sales.cloud` before returning the HTTP response to the client. Then, the client recognizes the domain name, and for the next request against *www1.sales.cloud* it attaches the cookie, which then successfully arrives at the server on **`machine1.intranet.corp`**.

> **i Note**
>
> Some Web servers use a syntax such as `domain=.intranet.corp` (RFC 2109), even though the newer RFC 6265 recommends using the notation without a dot.

> **i Note**
>
> The value of the domain attribute may be a simple host name, in which case no extra domain mapping is necessary on the Cloud Connector. If the server sets a cookie with `domain=machine1.intranet.corp`, the Cloud Connector automatically reverses the **`mapping machine1.intranet.corp`** to *www1.sales.cloud* and replaces the cookie domain accordingly.

Related Information

Configure Access Control [page 330]

1.4.5.2.10 Configure Solution Management Integration

Activate Solution Management reporting in the Cloud Connector.

If you want to monitor the Cloud Connector with the SAP Solution Manager, you can install a host agent on the machine of the Cloud Connector and register the Cloud Connector on your system.

Prerequisites

- You have installed the SAP Diagnostics Agent and SAP Host Agent on the Cloud Connector host and connected them to the SAP Solution Manager. As of Cloud Connector version 2.11.2, the RPM on Linux ensures that the host agent configuration is adjusted and that user groups are setup correctly.
  For more details about the host agent and diagnostics agent, see SAP Host Agent and the SCN Wiki SAP Solution Manager Setup/Managed System Checklist.
  See also SAP notes 2607632 (*SAP Solution Manager 7.2 - Managed System Configuration for SAP Cloud Connector*) and 1018839 (*Registering in the System Landscape Directory using sldreg*). For consulting, contact your local SAP partner.

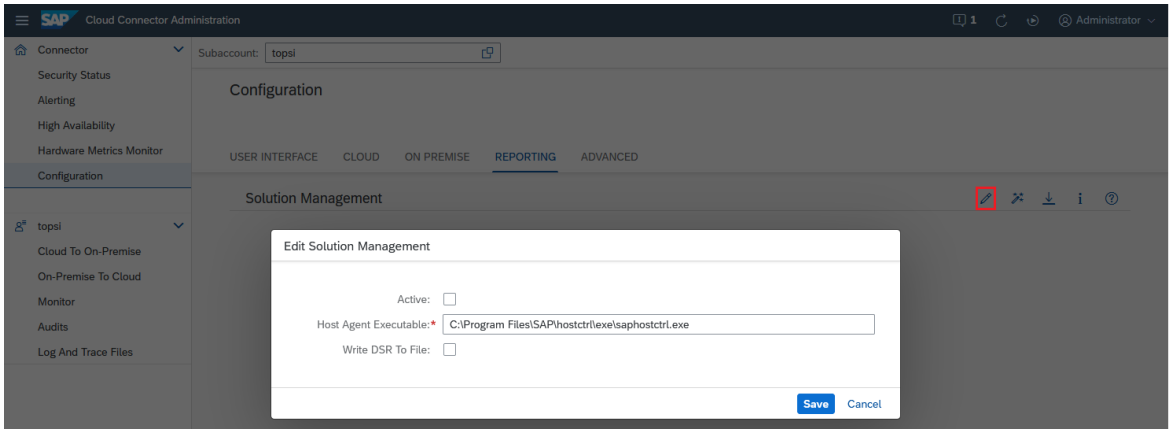
  > **i Note**
  >
  > Linux OS: if you installed the host agent after installing the Cloud Connector, you can execute `enableSolMan.sh` in the installation directory (available as of Cloud Connector version 2.11.2) to adjust the host agent configuration and user group setup. This action requires root permission.

- The SAP Solution Manager must be of release 7.2 SP06 or higher.

Procedure

To enable the integration, do the following:

1. From the Cloud Connector main menu, choose *Configuration* > *Reporting*. In section *Solution Management* of the *Reporting* tab, select *Edit*.



2. Select the *Active* checkbox.
3. In the field `<Host Agent>`, specify the location of the host agent as filepath.
4. If you want to store the reporting results (*Dynamic Statistical Records*), select *Write DSR To File*.
5. Choose *Save*.

> **i Note**
>
> To download the registration file *ImdbModel.xml*, choose the icon *Download registration file* from the *Reporting* tab.

Related Information

Monitoring [page 479]

1.4.5.2.11 Configure Tunnel Connections

Adapt connectivity settings that control the throughput by choosing the appropriate limits (maximal values).

If required, you can adjust the following parameters for the communication tunnel by changing their default values:

- Application Tunnel Connections (default: 1)

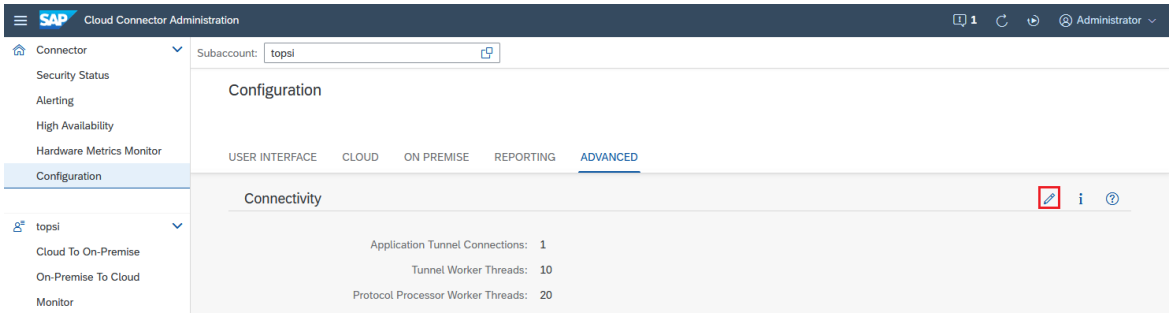
  > **i Note**
  >
  > This parameter specifies the default value for the maximal number of tunnel connections *per application*. The value must be higher than 0.

- Tunnel Worker Threads (default: 10)
- Protocol Processor Worker Threads (default: 20)

For detailed information on connection configuration requirements, see Configuration Setup [page 256].

To change the parameter values, do the following:

1. From the Cloud Connector main menu, choose *Configuration* > *Advanced*. In section *Connectivity*, select *Edit*.



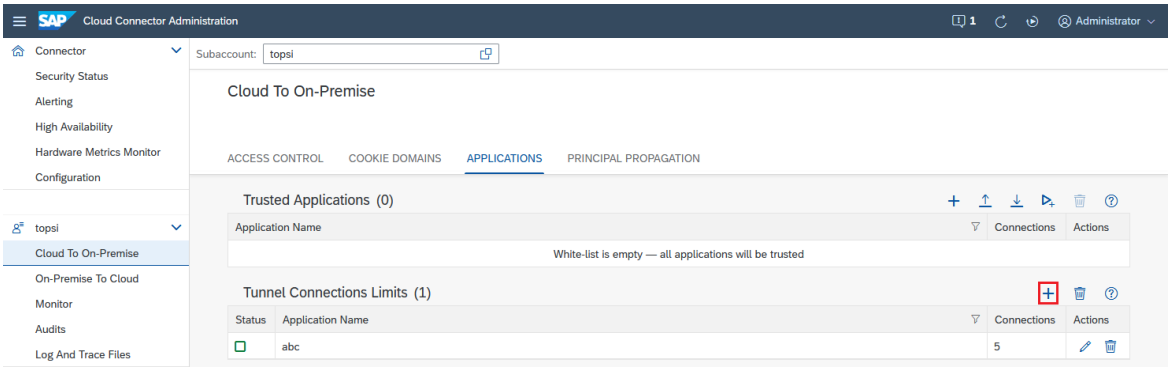
2. In the *Edit Connectivity Settings* dialog, change the parameter values as required.
3. Choose *Save*.

Additionally, you can specify the number of allowed tunnel connections for each application that you have specified as a trusted application [page 304].

> **i Note**
>
> If you don't change the value for a trusted application, it keeps the default setting specified above. If you change the value, it may be higher or lower than the default and must be higher than 0.

To add a specific connection limit to a trusted application, do the following:

1. From your subaccount menu, choose *Cloud To On-Premise* > *Applications*. In section *Tunnel Connection Limits*, choose *Add*.



2. In the *Edit Tunnel Connections Limit* dialog, enter the `<Application Name>` and change the number of `<Tunnel Connections>` as required.

> **i Note**
>
> The application name is visible in the SAP BTP cockpit under *Applications* > *Java Applications*. To allow a subscribed application, you must add it to the allowlist in the format **`<providerSubaccount>:<applicationName>`**. In particular, when using HTML5 applications, an implicit subscription to *services:dispatcher* is required.

3. Choose *Save*.

To edit this setting, select the application from the *Limits* list and choose *Edit*.

1.4.5.2.12 Configure the Java VM

Adapt the JVM settings that control memory management.

If required, you can adjust the following parameters for the Java VM by changing their default values:

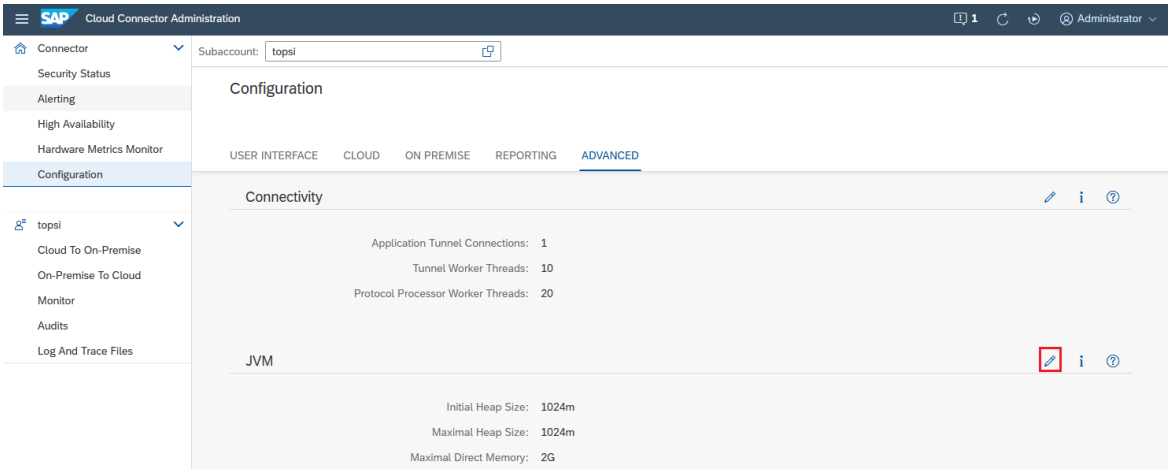
- Initial Heap Size (default: 1024 MB)
- Maximal Heap Size (default: 1024 MB)
- Maximal Direct Memory (default: 2 GB)

> **Note**
>
> A restart is required when changing JVM settings.

We recommended that you set the initial heap size equal to the maximal heap size, to avoid memory fragmentation.

To change the parameter values, do the following:

1. From the Cloud Connector main menu, choose *Configuration* > *Advanced*. In section *JVM*, select *Edit*.



2. In the *Edit JVM Settings* dialog, change the parameter values as required.
3. Choose *Save*.

1.4.5.2.13 Configuration Backup

You can backup and restore your Cloud Connector configuration.

To backup or restore your Cloud Connector configuration, do the following:

1. From the Cloud Connector main menu, choose *Connector*.



2. To backup or restore your configuration, choose the respective icon in the upper right corner of the screen.
   1. To backup your configuration, enter and repeat a password in the *Backup* dialog and choose *Backup*.

> **Note**
>
> An archive containing a snapshot of the current Cloud Connector configuration is created and downloaded by your browser. You can use this archive to restore the current state on this or a new Cloud Connector installation, if the original installation can no longer be used. *Do not* restore the backup on a second Cloud Connector, while the instance from which the backup was taken is *still active*. Such a setup might cause issues and is therefore not supported.

> For security reasons, some files are encrypted, using the password provided for the backup procedure.

2. To restore your configuration, enter the required *Archive Password* and the *Login Password* of the currently logged-in administrator user in the *Restore from Archive* dialog and choose *Restore*.

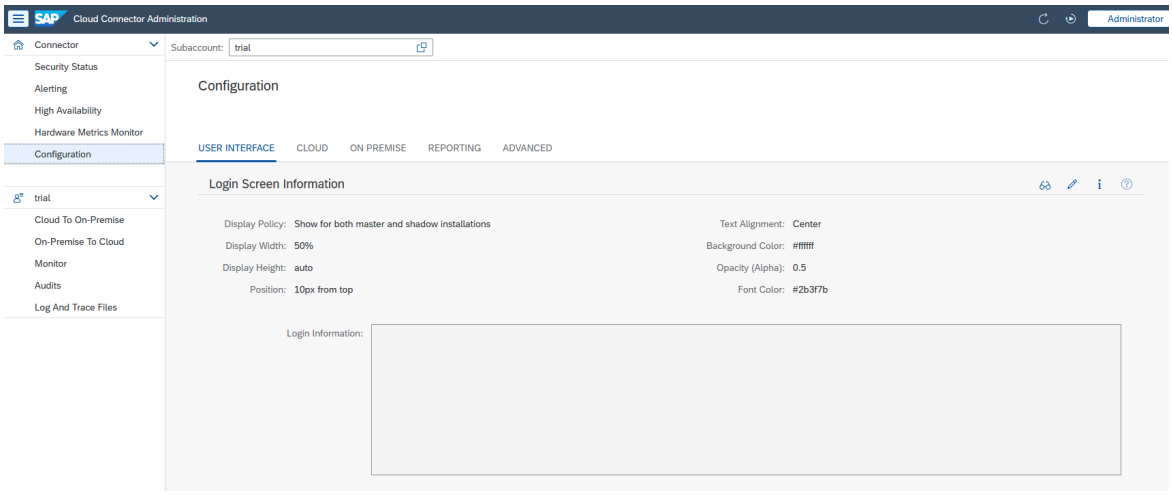
> **i Note**
>
> The restore action overwrites the current configuration of the Cloud Connector. It will be permanently lost unless you have created another backup before restoring. Upon successfully restoring the configuration, the Cloud Connector restarts automatically. All sessions are then terminated. The `props.ini` file however is treated in a special way. If the file in the backup differs from the one that is used in the current installation, it will be placed next to the original one as `props.ini.restored`. If you want to use the `props.ini.restored` file, replace the existing one on OS level and restart the Cloud Connector.

1.4.5.2.14 Configure Login Screen Information

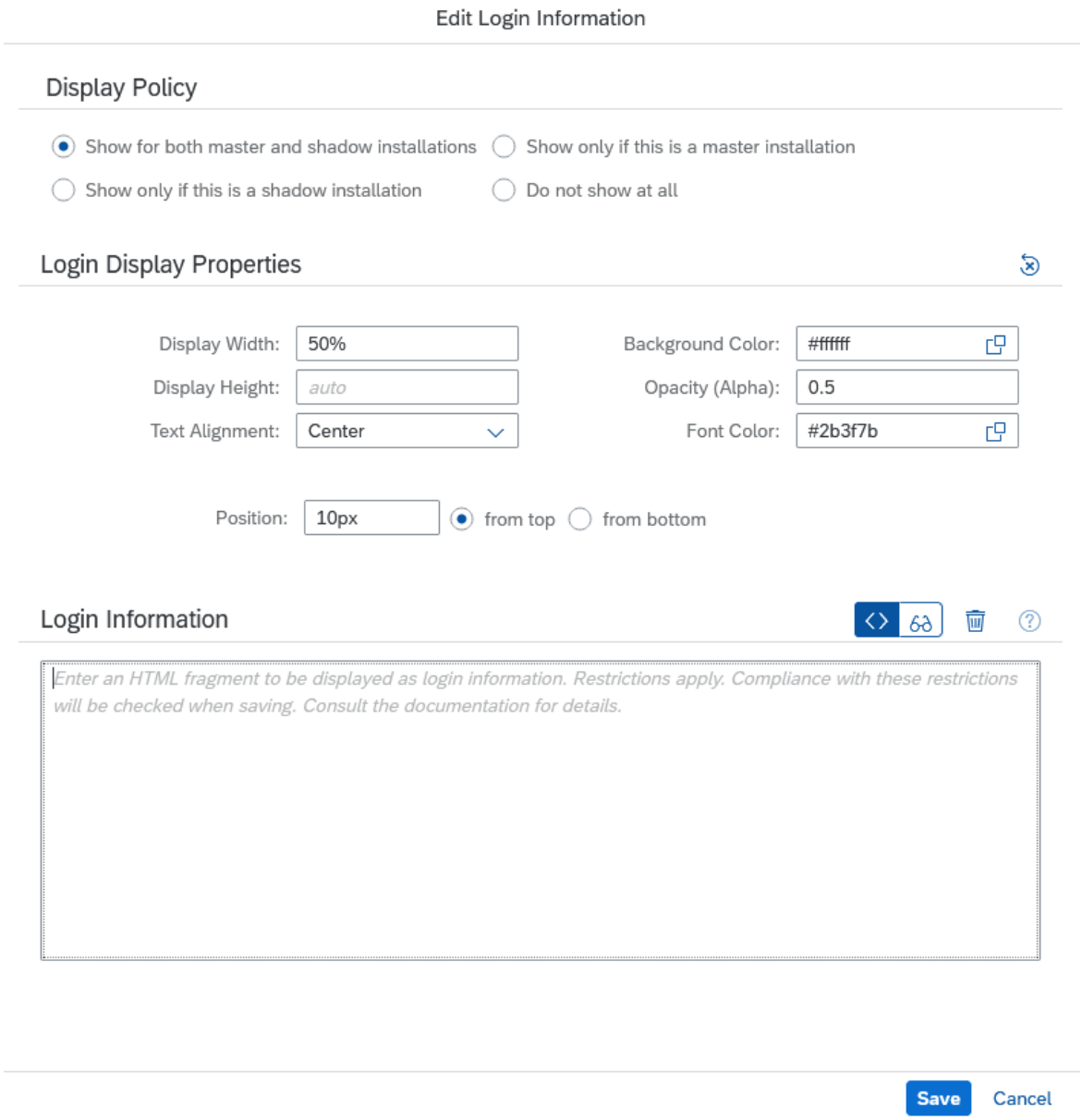
Add additional information to the login screen and configure its appearance.

To configure the login screen information, proceed as follows:

1. Go to *Configuration* > *User Interface* and press the *Edit* button in section *Login Screen Information*.



As a result, the following dialog is opened:



2. In section *Display Policy*, select a display policy for the login screen information.
3. In section *Display Properties*, specify your preferred display properties (appearance and position):
   - The login information is displayed in a box with rounded corners. You can specify its width and height in pixels (unit `px`) or as a percentage (unit `%`).

> **i Note**
>
> Omitting any value triggers the default or auto behavior.
>
> - For the width, the default behavior is equivalent to 100%.
> - For the height, the default behavior sets the height to a value that accommodates the login information (that is, the given HTML fragment). For extensive information we therefore recommend that you limit the height to a suitable pixel or percentage value to induce scrolling, and to prevent the box from growing beyond a reasonable height.

- When customizing the **background color** of the box, the **opacity** (of the background color), **font color**, or **text alignment**, then the section *Login Information* automatically switches to preview mode. That way you can follow live the changes made to the appearance of the box and of the information displayed inside it.

  > **i Note**
  >
  > You can hide the box and to show only the text of the login information by choosing an opacity value of **0** (opacity is the opposite of transparency. *No* opacity means *complete* transparency).

- You can **position the box** containing the login information at the top or bottom of the login page. To do this, set the field `<Position>` to the corresponding pixel or percentage value.

4. Enter the information to be displayed in section *Login Information*. The information must be supplied as an HTML fragment. There is a limited number of tags that can be used. Attributes available for these tags are subject to restrictions.

Available Tag	Attribute Restriction
p	Only attribute style is permitted, with property `text-align` set to a valid value (**`left`**, **`right`**, **`center`**, or **`justified`**)
ul	No attributes allowed
ol	No attributes allowed
li	No attributes allowed
br	No attributes allowed
h1	No attributes allowed
h2	No attributes allowed
h3	No attributes allowed
i	No attributes allowed
b	No attributes allowed
a	Must have exactly two attributes: ○ `href` (its value must be a **`<URL>`** with protocol **`http`** or **`https`**) ○ `target` (its value must be "**`_blank`**")

HTML syntax checking is strict. Attribute values must be enclosed by double quotes. Missing or unmatched opening or closing tags are not permitted.

> **i Note**
>
> Tag `br` does not require a closing tag as there cannot be any inner HTML.

1.4.5.3 Administration

Learn more about operating the Cloud Connector, using its administration tools and optimizing its functions.

Topic	Description
Configure Named Cloud Connector Users [page 465]	If you operate an LDAP server in your system landscape, you can configure the Cloud Connector to use the named users who are available on the LDAP server instead of the default Cloud Connector users.
High Availability Setup [page 470]	The Cloud Connector lets you install a redundant (shadow) instance, which monitors the main (master) instance.
Change the UI Port [page 476]	Use the changeport tool (Cloud Connector version 2.6.0+) to change the port for the Cloud Connector administration UI. .
Connect and Disconnect a Cloud Subaccount [page 476]	As a Cloud Connector administrator, you can connect the Cloud Connector to (and disconnect it from) the configured cloud subaccount.
Secure the Activation of Traffic Traces [page 477]	Tracing of network traffic data may contain business critical information or security sensitive data. You can implement a "four-eyes" (double check) principle to protect your traces (Cloud Connector version 1.3.2+).
Monitoring [page 479]	Use various views to monitor the activities and state of the Cloud Connector.
Alerting [page 505]	Configure the Cloud Connector to send email alerts whenever critical situations occur that may prevent it from operating.
Audit Logging [page 508]	Use the auditor tool to view and manage audit log information (Cloud Connector version 2.2+).
Troubleshooting [page 511]	Information about monitoring the state of open tunnel connections in the Cloud Connector. Display different types of logs and traces that can help you troubleshoot connection problems.
Process Guidelines for Hybrid Scenarios [page 516]	How to manage a hybrid scenario, in which applications running on SAP BTP require access to on-premise systems using the Cloud Connector.
Configuring Backup [page 519]	Find an overview of backup procedures for the Cloud Connector.

1.4.5.3.1 Configure Named Cloud Connector Users

Set up LDAP-based user management for the Cloud Connector.

LDAP-Based User Management

We recommend that you configure LDAP-based user management for the Cloud Connector to allow only named administrator users to log on to the administration UI.

This guarantees traceability of the Cloud Connector configuration changes via the Cloud Connector audit log. If you use the default and built-in `Administrator` user, you cannot identify the actual person or persons who perform configuration changes. Also, you will not be able to use different types of user groups.

Configuration

If you have an LDAP server in your landscape, you can configure the Cloud Connector to authenticate Cloud Connector users against the LDAP server.

Valid users or user groups must be assigned to one of the following roles:

- Administrator users: `admin` or `sccadmin`
- Display users: `sccdisplay` or `sccmonitoring`

  > **i Note**
  >
  > The role `sccmonitoring` provides access to the monitoring APIs, and is particularly used by the SAP Solution Manager infrastructure, see Monitoring APIs [page 487]. It cannot be used to access the Cloud Connector administation UI.

- Support users: `sccsupport`

Alternatively, you can define custom role names for each of these user groups, see: Use LDAP for Authentication [page 466].

Once configured, the default Cloud Connector `Administrator` user becomes inactive and can no longer be used to log on to the Cloud Connector.

1.4.5.3.1.1 Use LDAP for Authentication

You can use LDAP (Lightweight Directory Access Protocol) to configure Cloud Connector authentication.

After installation, the Cloud Connector uses file-based user management by default. Alternatively, the Cloud Connector also supports LDAP-based user management. If you operate an LDAP server in your landscape, you can configure the Cloud Connector to use the LDAP user base.

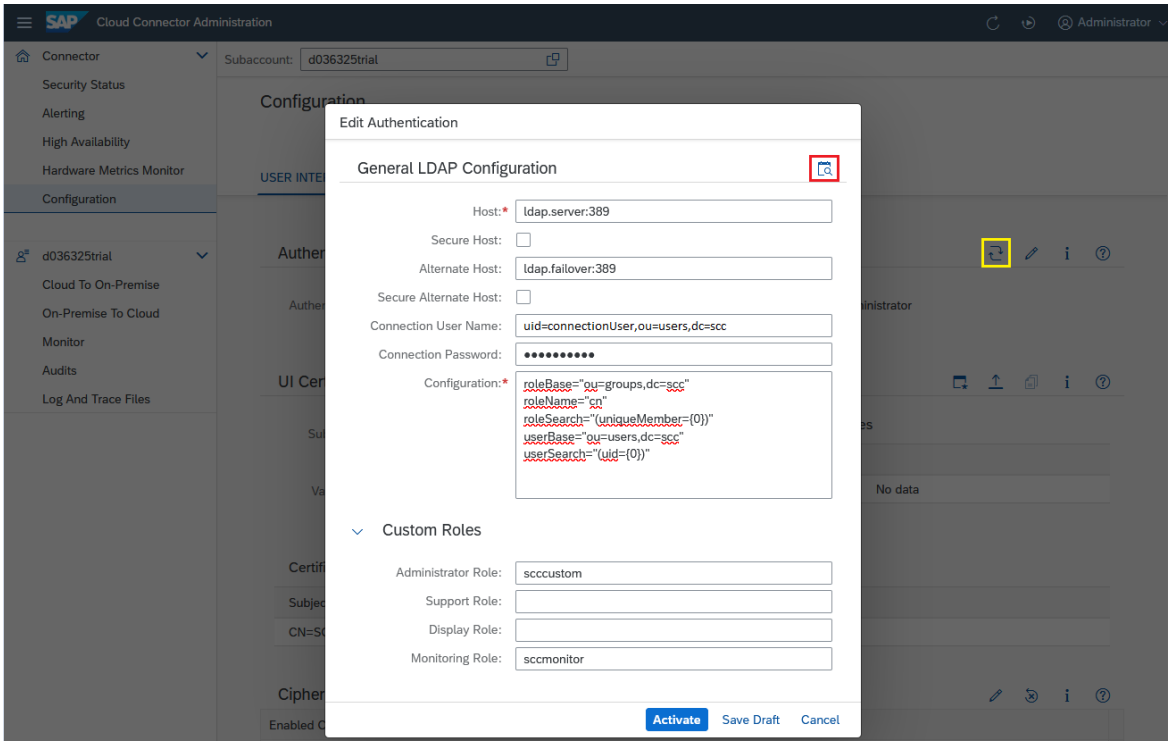
If LDAP authentication is active, you can assign users or user groups to the following default roles:

Default Role	Authorization
`sccadmin` or `admin`	Administrate the Cloud Connector (all CRUD operations).
`sccdisplay`	Access the Cloud Connector administration UI in read-only mode.
`sccsupport`	• Access the Cloud Connector administration UI in read-only mode. • Perform support-related tasks like setting trace levels or creating a thread dump.
`sccmonitoring`	Provides access to the monitoring APIs, and is particularly used by the SAP Solution Manager infrastructure, see Monitoring APIs [page 487]. **i Note** This role cannot be used to access the Cloud Connector administation UI.

Group membership is checked by the Cloud Connector.

Setting LDAP Authentication

1. From the main menu, choose *Configuration* and go to the *User Interface* tab.
2. From the *Authentication* section, choose *Switch to LDAP*.



3. (Optional) To save intermediate adoptions of the LDAP configuration, choose *Save Draft*. This lets you store the changes in the Cloud Connector without activation.
4. Usually, the LDAP server lists users in an LDAP node and user groups in another node. In this case, you can use the following template for LDAP configuration. Copy the template into the configuration text area:

```
roleBase="ou=groups,dc=scc"
roleName="cn"
roleSearch="(uniqueMember={0})"
userBase="ou=users,dc=scc"
userSearch="(uid={0})"
```

   Change the `<ou>` and `<dc>` fields in `userBase` and `roleBase`, according to the configuration on your LDAP server, or use some other LDAP query.

> **i Note**
>
> The configuration depends on your specific LDAP server. For details, contact your LDAP administrator.

5. Provide the LDAP server's host and port (port **389** is used by default) in the `<Host>` field. To use the secure protocol variant LDAPS based on TLS, select *Secure*.
6. Provide a failover LDAP server's host and port (port **389** is used by default) in the `<Alternate Host>` field. To use the secure protocol variant LDAPS based on TLS, select `<Secure Alternate Host>`.
7. (Optional) Depending on your LDAP server configuration you may need to specify the `<Connection User Name>` and its `<Connection Password>`. LDAP Servers supporting anonymous binding ignore these parameters.
8. (Optional) To use your own role names, you can customize the default role names in the *Custom Roles* section. If no custom role is provided, the Cloud Connector checks permissions for the corresponding default role name:
   - `<Administrator Role>` (default: `sccadmin`)

- `<Support Role>` (default: `sccsupport`)
- `<Display Role>` (default: `sccdisplay`)
- `<Monitoring Role>` (default: `sccmonitoring`)

9. (Optional) Before activating the LDAP authentication, you can execute an authentication test by choosing the *Test LDAP Configuration* button. In the pop-up dialog, you must specify user name and password of a user who is allowed to logon after activating the configuration. The check verifies if authentication would be successful or not.

> **i Note**
>
> We strongly recommend that you perform an authentication test. If authentication should fail, login is not possible anymore. The test dialog also provides a test protocol, which could be helpful for troubleshooting.

For more information about how to set up LDAP authentication, see tomcat.apache.org/tomcat-8.5-doc/realm-howto.html .

> **i Note**
>
> To find a list of all supported attributes, see https://tomcat.apache.org/tomcat-8.5-doc/config/realm.html#JNDI_Directory_Realm_-_org.apache.catalina.realm.JNDIRealm .

You can also configure LDAP authentication on the shadow instance in a high availability setup (master and shadow). From the main menu of the shadow instance, select *Shadow Configuration*, go to tab *User Interface*, and check the *Authentication* section.

> **i Note**
>
> If you are using LDAP together with a high availability setup, you cannot use the configuration option `userPattern`. Instead, use a combination of `userSearch`, `userSubtree` and `userBase`.

> **⚠ Caution**
>
> An LDAP connection over SSL/TLS can cause SSL errors if the LDAP server uses a certificate that is not signed by a trusted CA. If you cannot use a certificate signed by a trusted CA, you must set up the trust relationship manually, that is, import the public part of the issuer certificate to the JDK's trust storage.
>
> Usually, the *cacerts* file inside the java directory (`jre/lib/security/cacerts`) is used for trust storage. To import the certificate, you can use *keytool*:
>
> ```
> keytool -import -storepass changeit -file <certificate used by LDAP
> server> -keystore cacerts -alias <e.g. LDAP_xyz>
> ```
>
> For more information, see also https://docs.oracle.com/cd/E19830-01/819-4712/ablqw/index.html .

10. After finishing the configuration, choose *Activate*. Immediately after activating the LDAP configuration you must restart the Cloud Connector server, which invalidates the current browser session. Refresh the browser and logon to the Cloud Connector again, using the credentials configured at the LDAP server.
11. To switch back to file-based user management, choose the *Switch* icon again.

> **i Note**
>
> If you have set up an LDAP configuration incorrectly, you may not be able to logon to the Cloud Connector again. In this case, adjust the Cloud Connector configuration to use the file-based user store again *without the administration UI*. For more information, see the next section.

Switching Back to File-Based User Store without the Administration UI

If your LDAP settings do not work as expected, you can use the useFileUserStore tool, provided with Cloud Connector version 2.8.0 and higher, to revert back to the file-based user store:

1. Change to the installation directory of the Cloud Connector and enter the following command:
   - **Microsoft Windows**: `useFileUserStore`
   - **Linux, Mac OS**: `./useFileUserStore.sh`
2. Restart the Cloud Connector to activate the file-based user store.

For versions older than 2.8.0, you must manually edit the configuration files.

Depending on your operating system, the configuration file is located at:

- **Microsoft Windows OS**: `<install_dir>\config_master\org.eclipse.gemini.web.tomcat\default-server.xml`
- **Linux OS**: `/opt/sap/scc/config_master/org.eclipse.gemini.web.tomcat/default-server.xml`
- **Mac OS X**: `/opt/sap/scc/config_master/org.eclipse.gemini.web.tomcat/default-server.xml`

1. Replace the `Realm` section with the following:

```
<Realm className="org.apache.catalina.realm.LockOutRealm">
  <Realm className="org.apache.catalina.realm.CombinedRealm">
    <Realm
X509UsernameRetrieverClassName="com.sap.scc.tomcat.utils.SccX509SubjectDnRetri
ever" className="org.apache.catalina.realm.UserDatabaseRealm"
digest="SHA-256" resourceName="UserDatabase"/>
    <Realm
X509UsernameRetrieverClassName="com.sap.scc.tomcat.utils.SccX509SubjectDnRetri
ever" className="org.apache.catalina.realm.UserDatabaseRealm" digest="SHA-1"
resourceName="UserDatabase"/>
   </Realm>
</Realm>
```

2. Restart the Cloud Connector service:
   - **Microsoft Windows OS**: Open the Windows *Services* console and restart the `cloud connector` service.
   - **Linux OS**: Execute
     - System V init distributions: `service scc_daemon restart`
     - Systemd distributions: `systemctl restart scc_daemon`
   - **Mac OS X**: Not applicable because no daemon exists (for Mac OS X, only a portable variant is available).

1.4.5.3.2 High Availability Setup

You can operate the Cloud Connector in a high availability mode, in which a master and a shadow instance are installed.

Task	Description
Install a Failover Instance for High Availability [page 470]	Install a redundant Cloud Connector instance (shadow) that monitors the main instance (master).
Master and Shadow Administration [page 473]	Learn how to operate master and shadow instances.

Related Information

Connect DB Tools to SAP HANA via Service Channels [page 447]

1.4.5.3.2.1 Install a Failover Instance for High Availability

The Cloud Connector lets you install a redundant instance that monitors the main instance.

Context

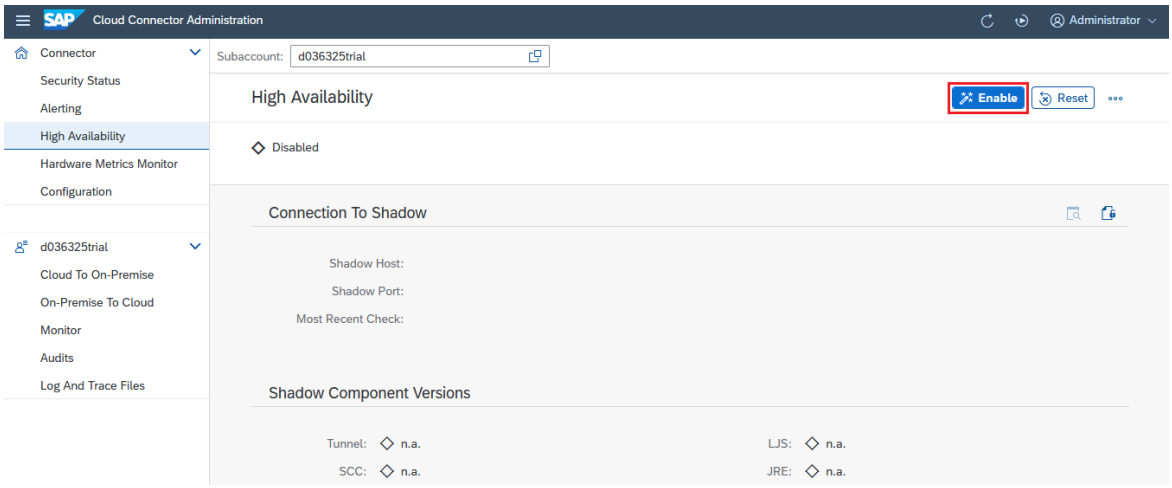
In a failover setup, when the main instance should go down for some reason, a redundant one can take over its role. The main instance of the Cloud Connector is called **master** and the redundant instance is called the **shadow**. The shadow has to be installed and connected to its master. During the setup of high availability, the master pushes the entire configuration to the shadow. Later on, during normal operation, the master also pushes configuration updates to the shadow. Thus, the shadow instance is kept synchronized with the master instance. The shadow pings the master regularly. If the master is not reachable for a while, the shadow tries to take over the master role and to establish the tunnel to SAP BTP.

> **i Note**
>
> For detailed information about sizing of the master and the shadow instance, see also Sizing Recommendations [page 252].

Procedure

Preparing the Master Instance for High Availability

1. Open the Cloud Connector UI and go to the master instance.
2. From the main menu, choose *High Availability*.
3. Choose *Enable*.



If this flag is not activated, no shadow instance can connect to this Cloud Connector. Additionally, when providing a concrete *Shadow Host*, you can ensure that only from this host a shadow instance can be connected.

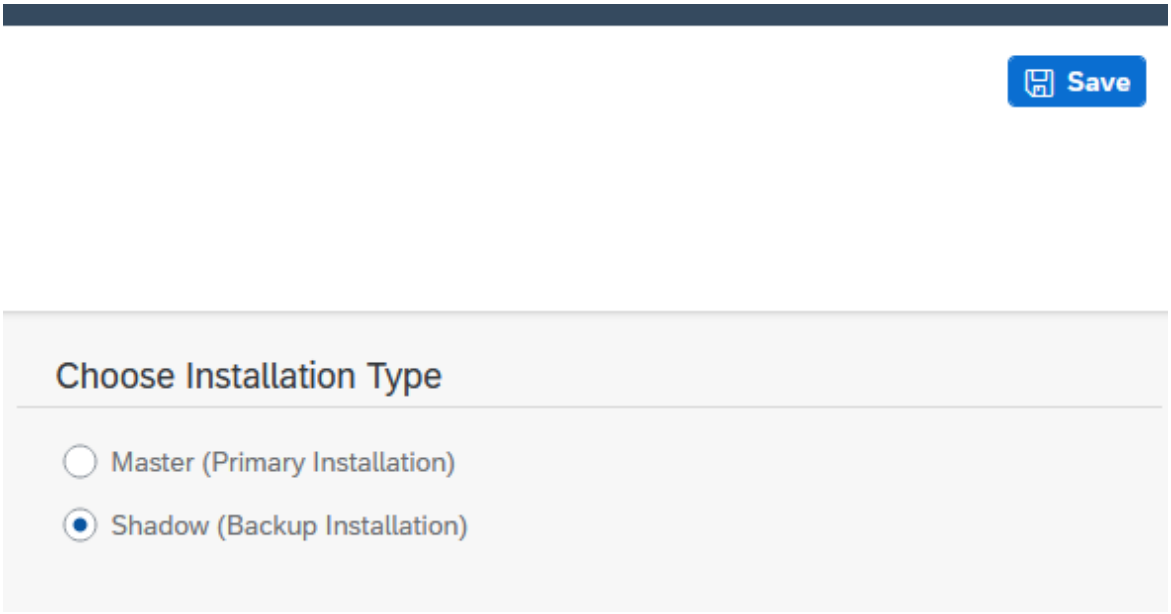
> ⚠ Caution
>
> Pressing the *Reset* button resets all high availability settings to their initial state. As a result, high availability is disabled and the shadow host is cleared. Reset only works if no shadow is connected.

Installing and Setting Up a Shadow Instance

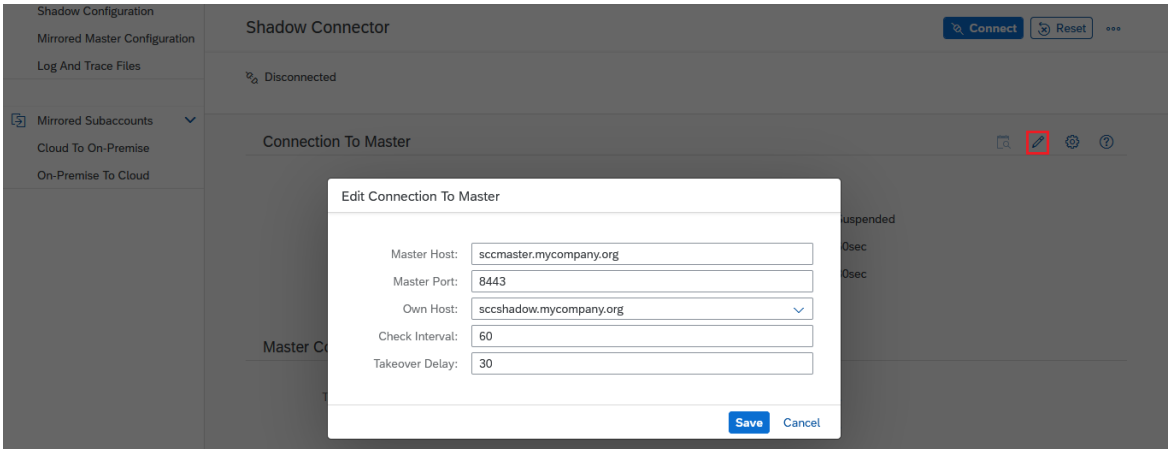
Install the shadow instance in the same network segment as the master instance. Communication between master and shadow via proxy is not supported. The same distribution package is used for master and shadow instance.

> ℹ Note
>
> If you plan to use LDAP for the user authentication on both master and shadow, make sure you configure it **before** you establish the connection from shadow to master.

1. On first start-up of a Cloud Connector instance, a UI wizard asks you whether the current instance should be master or shadow. Choose *Shadow* and *Save*:

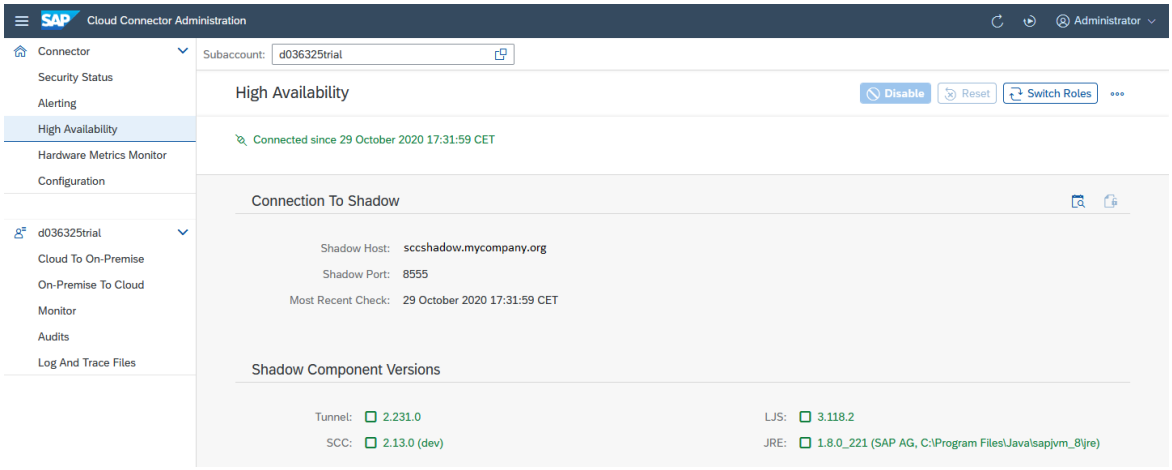


2. From the main menu, choose *Shadow Connector* and provide connection data for the master instance, that is, the master host and port. As of version 2.8.1.1, you can choose from the list of known host names, to use the host name under which the shadow host is visible to the master. You can specify a host name manually, if the one you want is not on the list. For the first connection, you must log on to the master instance, using the user name and password for the master instance. The master and shadow instances exchange X.509 certificates, which will be used for mutual authentication.



> **Note**
>
> If you want to attach the shadow instance to a different master, press the *Reset* button. All your high availability settings will be removed, that is, reset to their initial state. This works only if the shadow is not connected.

3. Upon a successful connection, the master instance pushes the entire configuration plus some information about itself to the shadow instance. You can see this information in the UI of the shadow instance, but you can't modify it.
4. The UI on the master instance shows information about the connected shadow instance. From the main menu, choose *High Availability*:



5. As of version **2.6.0**, the *High Availability* view includes an *Alert Messages* panel. It displays alerts if configuration changes have not been pushed successfully. This might happen, for example, if a temporary network failure occurs at the same time a configuration change is made. This panel lets an administrator know if there is an inconsistency in the configuration data between master and shadow that could cause trouble if the shadow needs to take over. Typically, the master recognizes this situation and tries to push the configuration change at a later time automatically. If this is successful, all failure alerts are removed and replaced by a warning alert showing that there had been trouble before. As of version **2.8.0.1**, these alerts have been integrated in the general *Alerting* section; there is no longer a separate *Alert Messages* panel.
   If the master doesn't recover automatically, disconnect, then reconnect the shadow, which triggers a complete configuration transfer.

Related Information

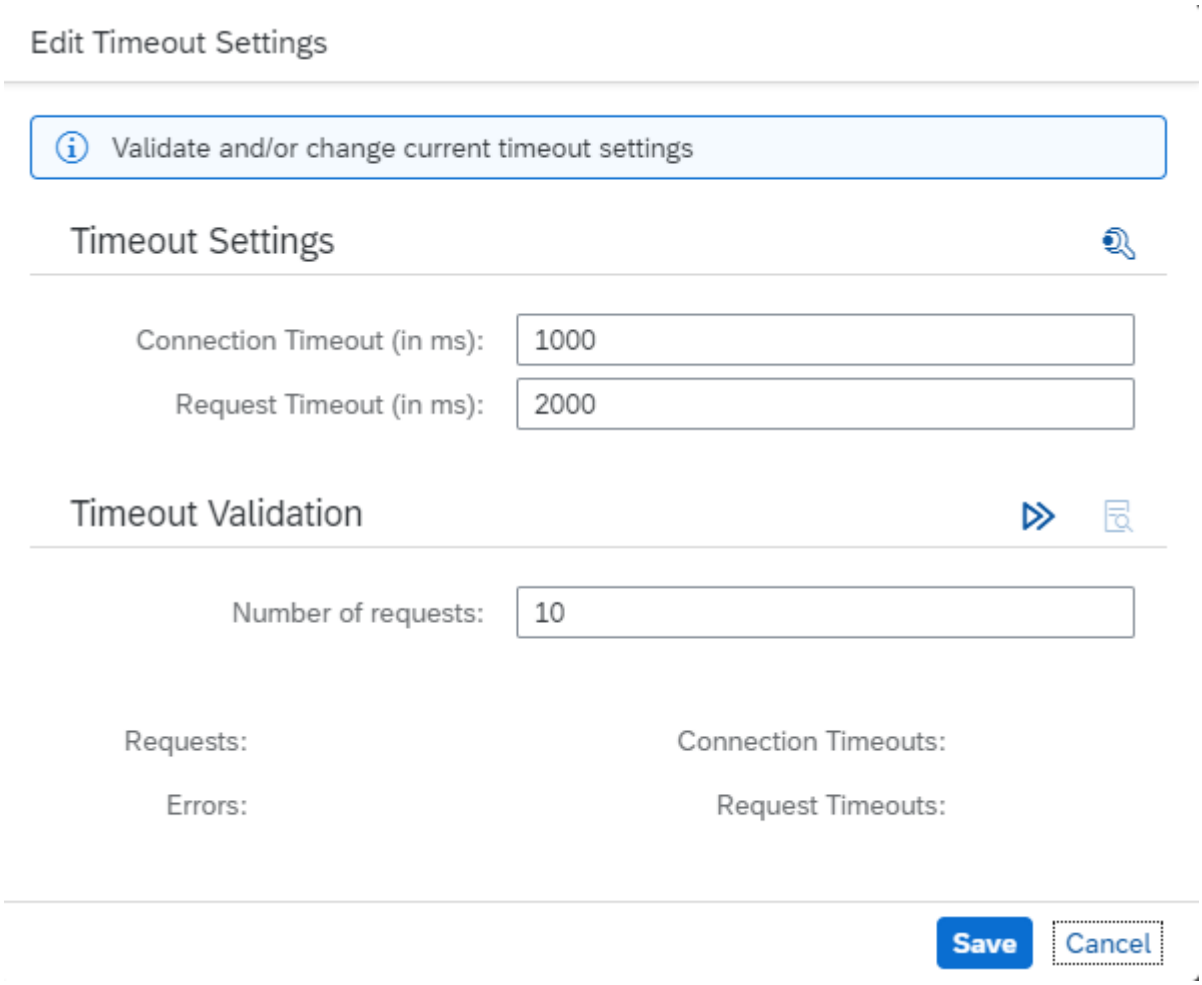
Initial Configuration [page 278]
Master and Shadow Administration [page 473]

1.4.5.3.2.2 Master and Shadow Administration

Administration of Shadow Instances

There are several administration activities you can perform on the shadow instance. All configuration of tunnel connections, host mappings, access rules, and so on, must be maintained on the master instance; however, you can replicate them to the shadow instance for display purposes. You may want to modify the **check interval** (time between checks of whether the master is still alive) and the **takeover delay** (time the shadow waits to see whether the master would come back online, before taking over the master role itself).

As of Cloud Connector version 2.11.2, you can configure the timeout for the connection check, by pressing the gear icon in the section *Connection To Master* of the shadow connector main page.



The `<Connection Timeout>` field specifies the maximum time allowed for establishing the technical connection to the master, the `<Request Timeout>` defines the maximum time allowed for executing the check over this connection. In the *Timeout Validation* section, you can check the current settings by pressing the *Execute* button. If the timeouts are hit, you might want to increase the settings accordingly.

Keep in mind the following points:

- The log level on master and shadow instances can be different.
- Configuration for check interval and takeover delay is maintained only on the shadow instance, and is transferred to the master for display purposes.
- Audit logs are only written on the master instance and are not transferred to the shadow. However, if the shadow becomes the master for some time, the audit log is potentially distributed over both master and shadow instances.

You can use the *Reset* button to drop all the configuration information on the shadow that is related to the master, but only if the shadow is not connected to the master.

Required Configuration on the Shadow Instance

Once connected to the master, the shadow instance receives the configuration from the master instance. Yet, there are some aspects you must configure on the shadow instance separately:

- **User administration** is configured separately on master and shadow instances. Generally, it is not required to have the same configuration on both instances. In most cases, however, it is suitable to configure master and shadow in the same way.
- The **UI certificate** is not shared. Each host can have its own certificate, so you must maintain the UI certificates on master and shadow. You can use the same certificate though.
- **SNC configuration**: If secure RFC communication or principal propagation for RFC calls is used, you must configure SNC on each instance separately.

Failover Process

The shadow instance regularly checks whether the master instance is still alive. If a check fails, the shadow instance first attempts to reestablish the connection to the master instance for the time period specified by the takeover delay parameter.

- If no connection becomes possible during the takeover delay time period, the shadow tries to take over the master role. At this point, it is still possible for the master to be alive and the trouble to be caused by a network issue between the shadow and master. The shadow instance next attempts to establish a tunnel to the given SAP BTP subaccount. If the original master is still alive (that is, its tunnel to the cloud subaccount is still active), this attempt is denied and the shadow instance remains in "shadow status", periodically pinging the master and trying to connect to the cloud, while the master is not yet reachable.
- If the takeover delay period has fully elapsed, and the shadow instance does make a connection, the cloud side opens a tunnel and the shadow instance takes over the role of the master. From this point, the shadow instance shows the UI of the master instance and allows the usual operations of a master instance, for example, starting/stopping tunnels, modifying the configuration, and so on.

When the original master instance restarts, it first checks whether the registered shadow instance has taken over the master role. If it has, the master registers itself as a shadow instance on the former shadow (now master) instance. Thus, the two Cloud Connector installations, in fact, have switched their roles.

> **i Note**
>
> Only one shadow instance is supported. Any further shadow instances that attempt to connect are declined by the master instance.

The master considers a shadow as lost, if no check/ping is received from that shadow instance during a time interval that is equal to three times the check period. Only after this much time has elapsed can another shadow system register itself.

> **i Note**
>
> On the master, you can manually trigger failover by selecting the *Switch Roles* button. If the shadow is available, the switch is made as expected. Even if the shadow instance cannot be reached, the role switch of

the master may still be enforced. Select *Switch Roles* only if you are absolutely certain it is the correct action to take for your current circumstances.

1.4.5.3.3 Change the UI Port

Context

By default, the Cloud Connector uses port **8443** for its administration UI. If this port is blocked by another process, or if you want to change it after the installation, you can use the `changeport` tool, provided with Cloud Connector version **2.6.0** and higher.

> **i Note**
>
> On Windows, you can also choose a different port during installation.

Procedure

1. Change to the installation directory of the Cloud Connector. To adjust the port and execute one of the following commands:
   - Microsoft Windows OS:

     ```
     changeport <desired_port>
     ```

   - Linux OS, Mac OS X:

     ```
     ./changeport.sh <desired_port>
     ```

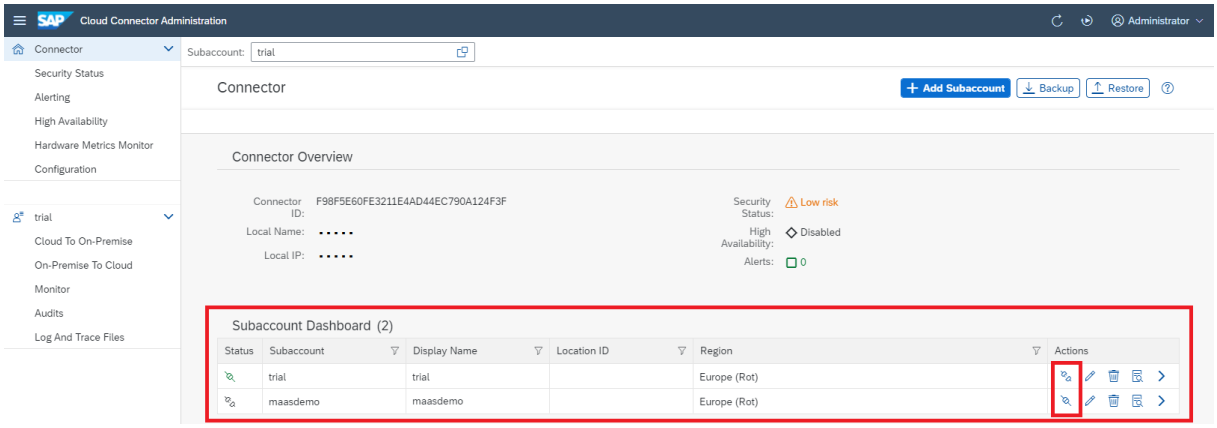
2. When you see a message stating that the port has been successfully modified, restart the Cloud Connector to activate the new port.

1.4.5.3.4 Connect and Disconnect a Cloud Subaccount

The major principle for the connectivity established by the Cloud Connector is that the Cloud Connector administrator should have full control over the connection to the cloud, that is, deciding if and when the Cloud Connector should be connected to the cloud, the accounts to which it should be connected, and which on-premise systems and resources should be accessible to applications of the connected subaccount.

Using the administration UI, the Cloud Connector administrator can connect and disconnect the Cloud Connector to and from the configured cloud subaccount. Once disconnected, no communication is possible, either between the cloud subaccount and the Cloud Connector, or to the internal systems. The connection

state can be verified and changed by the Cloud Connector administrator on the Subaccount Dashboard tab of the UI.



> **i Note**
>
> Once the Cloud Connector is freshly installed and connected to a cloud subaccount, none of the systems in the customer network are yet accessible to the applications of the related cloud subaccount. Accessible systems and resouurces must be configured explicitly in the Cloud Connector one by one, see Configure Access Control [page 330].

A Cloud Connector instance can be connected to multiple subaccounts in the cloud. This is useful especially if you need multiple subaccounts to structure your development or to stage your cloud landscape into development, test, and production. In this case, you can use a single Cloud Connector instance for multiple subaccounts. However, we recommend that you do not use subaccounts running in productive scenarios and subaccounts used for development or test purposes within the same Cloud Connector. You can add or a delete a cloud account to or from a Cloud Connector using the *Add* and *Delete* buttons on the *Subaccount Dashboard* (see screenshot above).

Related Information

Managing Subaccounts [page 289]

1.4.5.3.5 Secure the Activation of Traffic Traces

For support purposes, you can trace HTTP and RFC network traffic that passes through the Cloud Connector.

Context

Traffic data may include business-critical information or security-sensitive data, such as user names, passwords, address data, credit card numbers, and so on. Thus, by activating the corresponding trace level, a

Cloud Connector administrator might see data that he or she is not meant to. To prevent this behavior, implement the four-eyes principle, which is supported by the Cloud Connector release **1.3.2** and higher.

Once the four-eyes principle is applied, activating a trace level that dumps traffic data will require two separate users:

- An operating system user on the machine where the Cloud Connector is installed;
- An `Administrator` user of the Cloud Connector user interface.

By assigning these roles to two different people, you can ensure that both persons are needed to activate a traffic dump.

Four-Eyes Principle for Microsoft Windows OS

1. Create a file named **writeHexDump** in `<scc_install_dir>\scc_config`. The owner of this file must be a user other than the operating system user who runs the `cloud connector` process.

   > **i** Note
   >
   > Usually, this file owner is the user which is specified in the *Log On* tab in the properties of the `cloud connector` service (in the Windows *Services* console). We recommend that you use a dedicated OS user for the `cloud connector` service.

   - Only the file owner should have write permission for the file.
   - The OS user who runs the `cloud connector` process needs read-only permissions for this file.
   - Initially, the file should contain a line like `allowed=false`.
   - In the security properties of the file `scc_config.ini` (same directory), make sure that only the OS user who runs the `cloud connector` process has write/modify permissions for this file. The most efficient way to do this is simply by removing all other users from the list.
2. Once you've created this file, the Cloud Connector refuses any attempt to activate the *Payload Trace* flag.
3. To activate the payload trace, first the owner of `writeHexDump` must change the file content from `allowed=false` to `allowed=true`. Thereafter, the *Administrator* user can activate the payload trace from the Cloud Connector administration screens.

Four-Eyes Principle for Linux OS/Mac OS X

1. Create a file named **writeHexDump** in `/usr/local/vl/base/cfg` (Cloud Connector 1.3.2) or `/opt/sap/scc/scc_config` (Cloud Connector 2.x). The owner of this file must be a user other than the **`scctunnel`** user (that is, the operating system user under which the `cloud connector` processes run) and not a member of the operating system user group **`sccgroup`**.
   - Only the file owner should have write permission for the file.
   - The **`scctunnel`** user needs read-only permissions for this file.
   - Initially, the file should contain a line like `allowed=false`.
2. Once you've created this file, the Cloud Connector refuses any attempt to set the trace level higher than `Runtime` (Cloud Connector 1.3.2) or to activate the *Payload Trace* flag (Cloud Connector 2.x).

3. To set a higher trace level, which includes traffic Hex-dumps (Cloud Connector 1.3.2), or to activate the payload trace (Cloud Connector 2.x), the file owner mentioned above must first change the file content from `allowed=false` to `allowed=true`. Thereafter, the *Administrator* user can activate one of the higher trace levels (Cloud Connector 1.3.2) or the payload trace (Cloud Connector 2.x) from the Cloud Connector administration screens.

1.4.5.3.6 Monitoring

Learn how to monitor the Cloud Connector from the SAP BTP cockpit and from the Cloud Connector administration UI.

Checking the Operational State

The simplest way to verify whether a Cloud Connector is running is to try to access its administration UI. If you can open the UI in a Web browser, the `cloud connector` process is running.

- On Microsoft Windows operating systems, the `cloud connector` process is registered as a Windows service, which is configured to start automatically after a new Cloud Connector installation. If the Cloud Connector server is rebooted, the `cloud connector` process should also auto-restart immediately. You can check the state with the following command:

  ```
  sc query "SAP Cloud Connector"
  ```

  The line state shows the state of the service.
- On Linux operating systems, the Cloud Connector is registered as a daemon process and restarts automatically each time the `cloud connector` process is down, for example, following a system restart. You can check the daemon state with the following command:

  ```
  service scc_daemon status
  ```

To verify if a Cloud Connector is connected to a certain cloud subaccount, log on to the Cloud Connector administration UI and go to the *Subaccount Dashboard*, where the connection state of the connected subaccounts is visible, as described in section Connect and Disconnect a Cloud Subaccount [page 476].

Monitoring from the Cockpit

The cockpit includes a *Connectivity* section, where users can check the status of the Cloud Connector(s) attached in the current subaccount, if any, as well as information about the Cloud Connector ID, version, used Java runtime, high availability setup (master and shadow instance), and so on (choose *Connectivity* > *Cloud Connectors*).

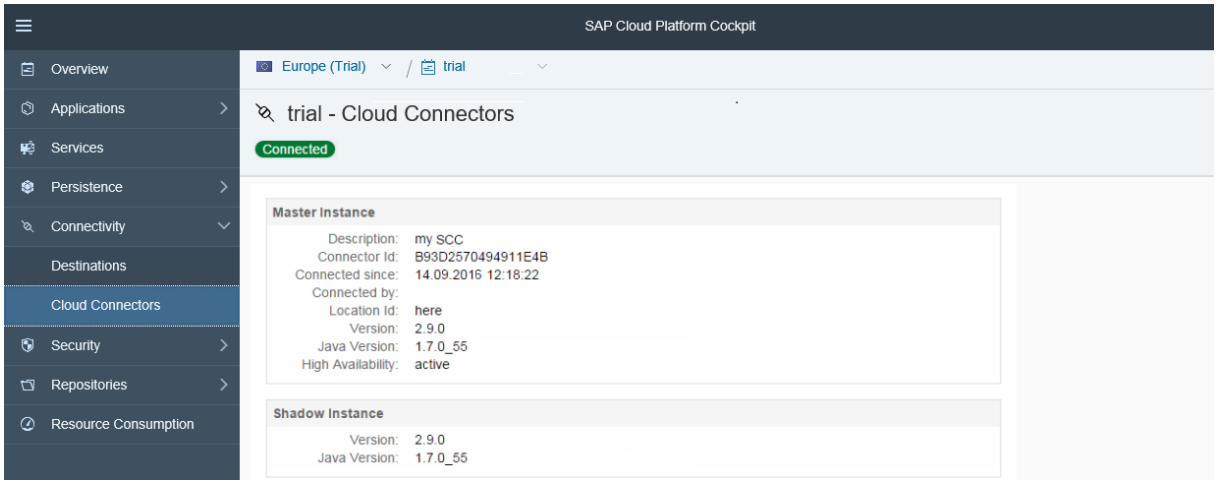
Access to this view is granted to:

- Neo envirnoment: Users with a role containing the permission `readSCCTunnels`, for example, the predefined role `Cloud Connector Admin`.
- Cloud Foundry environment, feature set A: Users with a Cloud Foundry org role containing the permission `readSCCTunnels`, for example, the role `Org Manager`.

  > **i Note**
  >
  > As a prerequisite, a Cloud Foundry org must be available.

- Cloud Foundry environment, feature set B: Users with a role containing the permission `readSCCTunnels`, for example, the predefined role `Cloud Connector Administrator`.

> **i Note**
>
> For more information on feature sets in the Cloud Foundry environment, see Cloud Management Tools — Feature Set Overview.



Monitoring from the Cloud Connector Amination UI

The Cloud Connector offers various views for monitoring its activities and state.

You can check the overall state of the Cloud Connector through its Hardware Metrics [page 481], whereas subaccount-specific performance and usage data is available via Subaccount-Specific Monitoring [page 482]. To provide external monitoring tools, you can use the Monitoring APIs [page 487].

Related Information

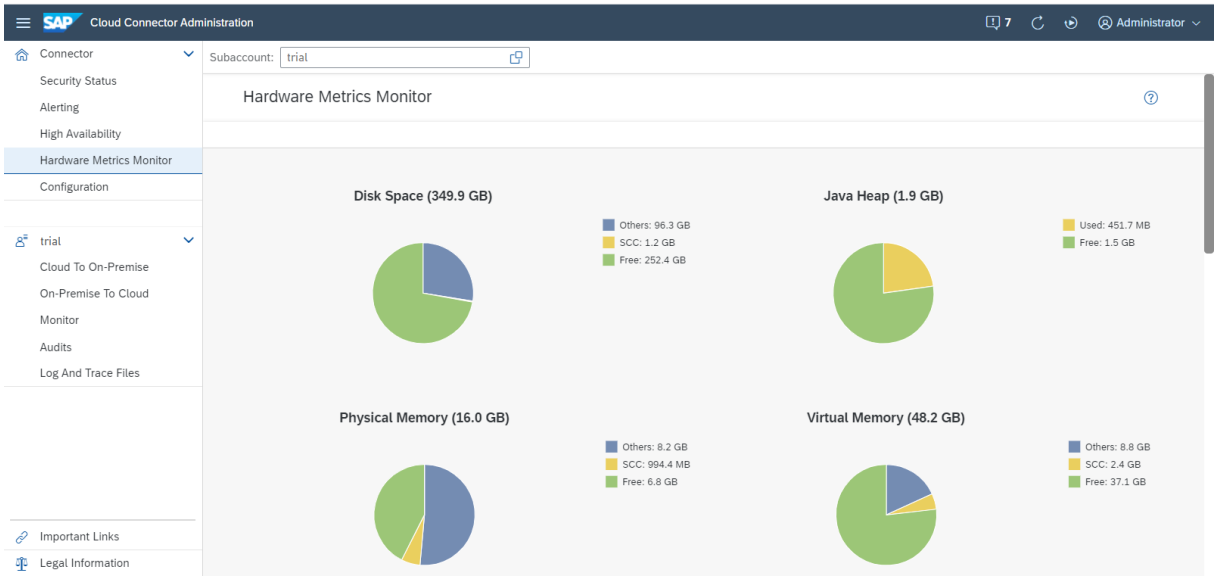
Configure Solution Management Integration [page 457]
High Availability Setup [page 470]

1.4.5.3.6.1 Hardware Metrics

Check the current state of critical system resources in the Cloud Connector.

You can check the current state of critical system resources (disc space, Java heap, physical memory, virtual memory) using pie charts.

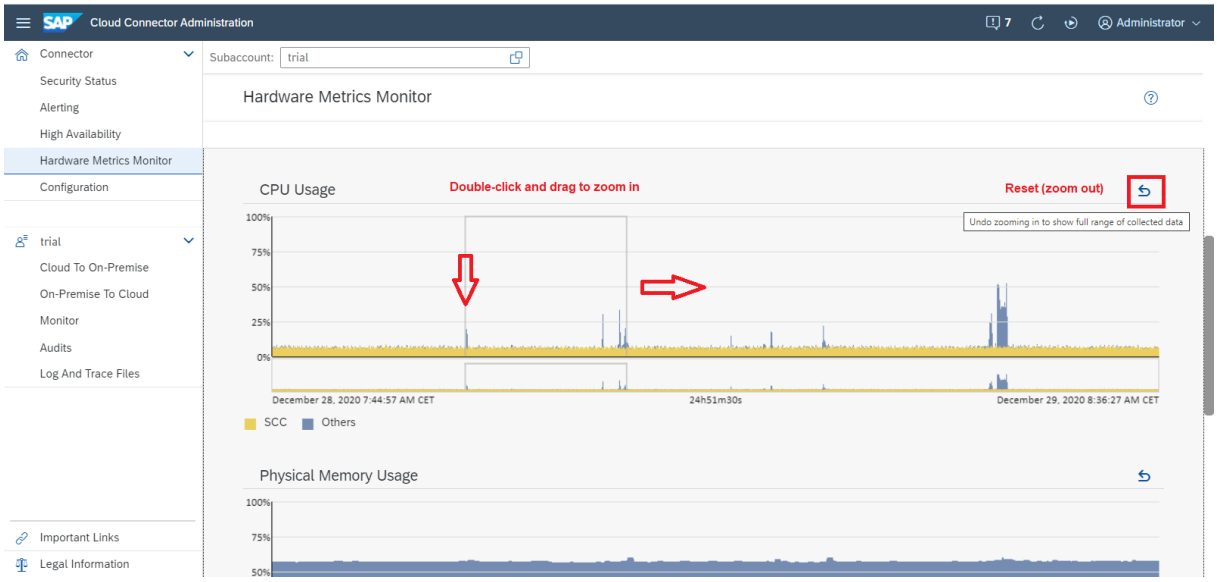
To access the monitor, choose *Hardware Metrics Monitor* from the main menu.



In addition, the history of CPU and memory usage (physical memory, Java heap) is shown in history graphs below the pie charts (recorded in intervals of 15 seconds).

You can view the usage data for a selected time period in each history graph:

- Double-click inside the main graph area to set the start (or end) point, and drag to the left or to the right to zoom in.
  - The entire timeline is always visible in the smaller bottom area right below the main graph.
  - A frame in the bottom area shows the position of the selected section in the overall timeline.
- Choose *Undo zooming in...* to reset the main graph area to the full range of available data.



1.4.5.3.6.2 Subaccount-Specific Monitoring

Use different monitoring views in the Cloud Connector administration UI to check subaccount-specific activites and data.

Content

Performance Overview [page 482]

Most Recent Requests [page 483]

Resource Filter Settings [page 484]

Top Time Consumers [page 485]

Usage Statistics [page 485]

Backend Connections [page 486]

Performance Overview

All requests that travel through the Cloud Connector to a backend system, as specified through access control, take a certain amount of time. You can check the duration of requests in a bar chart. The requests are not shown individually, but are assigned to buckets, each of which represents a time range.

For example, the first bucket contains all requests that took 10ms or less, the second one the requests that took longer than 10ms, but not longer than 20ms. The last bucket contains all requests that took longer than 5000ms.

In case of latency gaps, you may try to adjust the influencing parameters: number of connections, tunnel worker threads, and protocol processor worker threads. For more information, see Configuration Setup [page 256].

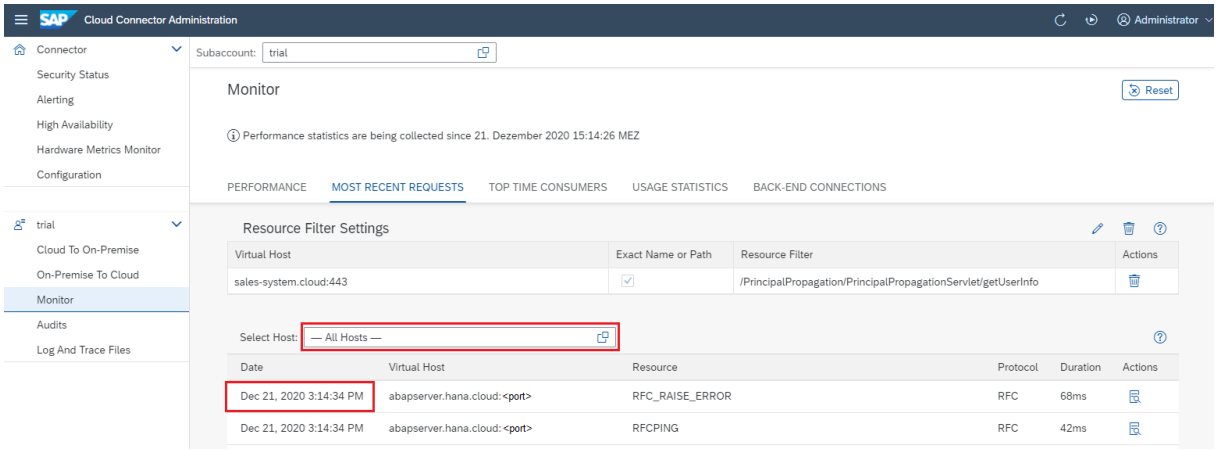
The collection of duration statistics starts as soon as the Cloud Connector is operational. You can delete all of these statistical records by selecting the button *Delete All*. After that, the collection of duration statistics starts over.

> **i Note**
>
> *Delete All* deletes not only the list of most recent requests, but it also clears the top time consumers.

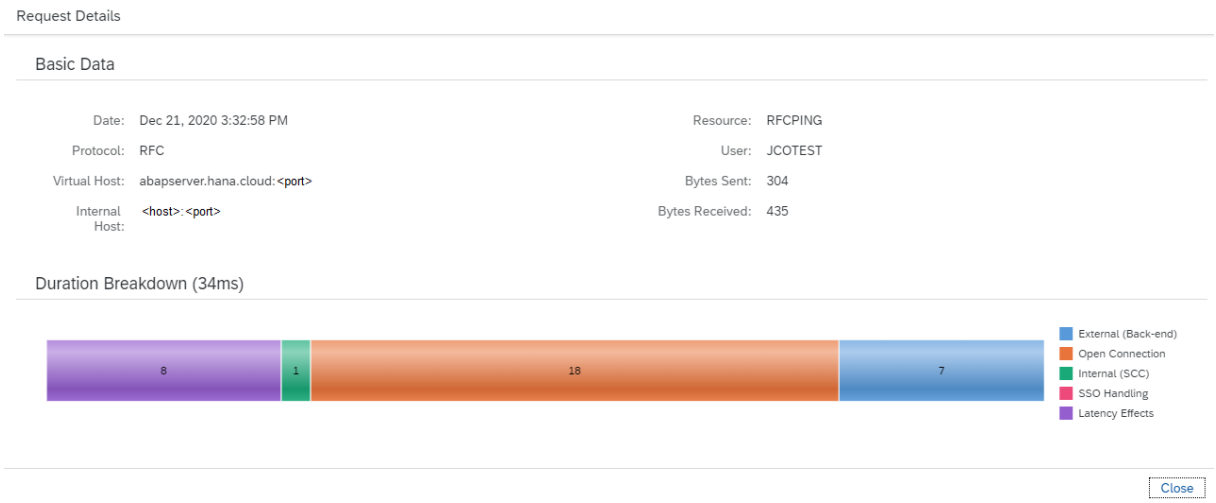
Back to Content [page 482]

Most Recent Requests

This option shows the most recent requests:



The number of requests that are shown is limited to 50. You can either view all requests or only the ones destined for a certain virtual host, which you can select.You can select a row to see more detail.



A horizontal stacked bar chart breaks down the duration of the request into several parts: external (backend), open connection, internal (SCC), SSO handling, and latency effects. The numbers in each part represent milliseconds.

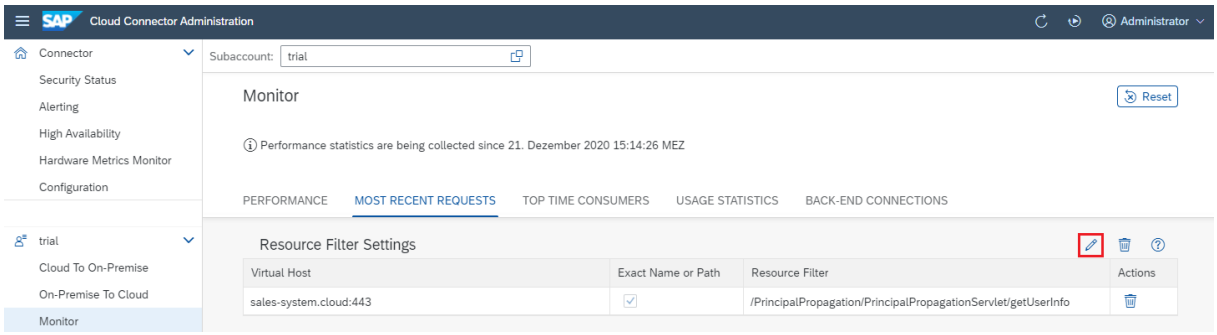
> **i Note**
>
> Parts with a duration of less than 1ms are not included.

In the above example, the selected request took 34ms, to which the Cloud Connector contributed 1ms. Opening a connection took 18ms. Backend processing consumed 7ms. Latency effects accounted for the remaining 8ms, while there was no SSO handling necessary and hence it took no time at all.

Back to Content [page 482]

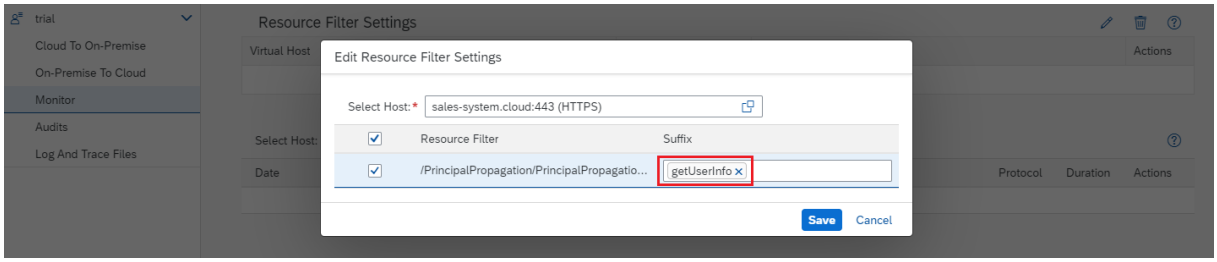
Resource Filter Settings

To further restrict the selection of the listed 50 most recent requests, you can edit the resource filter settings for each virtual host:



In the *Edit* dialog, select the virtual host for which you want to specify the resource filter and choose one or more of the listed accessible resources. This list includes all resources that have been exposed during access

control configuration (see also: Configure Access Control [page 330]). If the access policy for an accessible resource is set to `Path and all sub-paths`, you can further narrow the selection by adding one or more sub-paths to the resource as a suffix .



Each selected resource/sub-path is listed separately in the resource filter list.

> **i Note**
>
> If you specify sub-paths for a resource, the request URL must match exactly one of these entries to be recorded. Without specified sub-paths (and the value `Path and all sub-paths` set for a resource), all sub-paths of a specified resource are recorded.

Back to Content [page 482]

Top Time Consumers

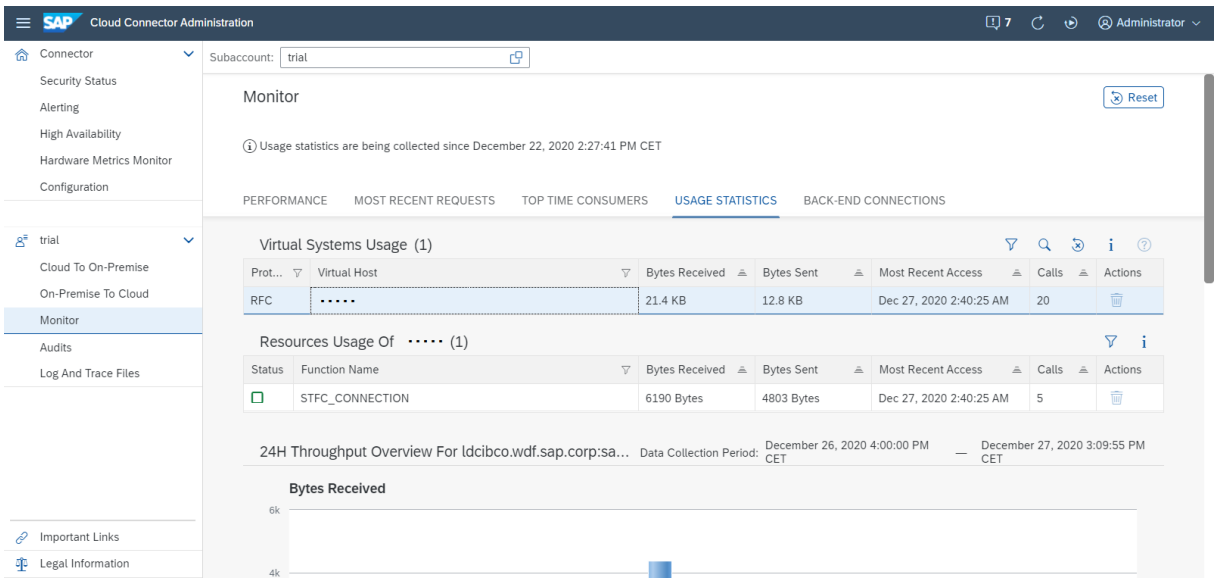
This option is similar to *Most Recent Requests*; however, requests are not shown in order of appearance, but rather sorted by their duration (in descending order). Furthermore, you can delete top time consumers, which has no effect on most recent requests or the performance overview.

Back to Content [page 482]

Usage Statistics

To view the statistical data regarding the traffic handled by each **virtual host**, you can select a virtual host from the table. The detail view shows the traffic handled by *each resource*, as well as a *24 hour overview* of the throughput as a bar chart that aggregates the throughput (bytes received and bytes sent by a virtual host, respectively) on an hourly basis.

> **i Note**
>
> Currently, this feature is only available for the protocols HTTP(S) and RFC (SNC). Virtual hosts using other protocols are not listed.

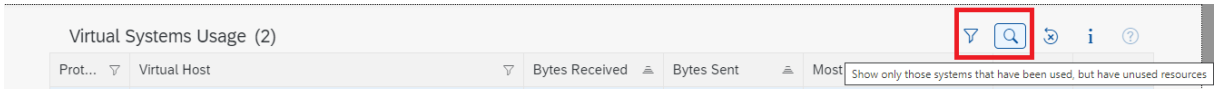


The data that is collected includes the number of bytes received *from* cloud applications and the number of bytes sent back *to* cloud applications. The time of the most recent access is shown for the virtual hosts, and in the detail view also for the resources. If no access has taken place yet, the most recent access is shown as `n.a.` (not available). Similarly, the number of bytes received and sent of a virtual host is the sum of bytes received and sent of its resources.

The tables listing usage statistics of virtual hosts and their resources let you delete unused virtual hosts or unused resources. Use action *Delete* to delete such a virtual host or resource.

> ⚠ Caution
>
> - Usage statistics are collected during runtime only and are not stored when stopping the Cloud Connector. That is, these statistics are lost when the Cloud Connector is stopped or restarted.
> - Use care when taking the decision to delete a resource or virtual host based on its usage statistics.

For both virtual hosts and resources, you can use a classic filter button to reduce the virtual hosts or resources to those that have never been used (since the Cloud Connector started). For the virtual hosts, a second filter type is available that selects only those virtual hosts that have been used, but *include resources never used*. This feature facilitates locating obsolete resources of otherwise active virtual hosts.



Back to Content [page 482]

Backend Connections

This option shows a tabular overview of all active and idle connections, aggregated for each virtual host. By selecting a row (each of which represents a virtual host) you can view the details of all active connections as

well as a graphical summary of all idle connections. The graphical summary is an accumulative view of connections based on the time the connections have been idle.

The maximum idle time appears on the rightmost side of the horizontal axis. For any point `t` on that axis (representing a time value ranging between 0ms and the maximal idle time), the ordinate is the number of connections that have been idle for no longer than `t`. You can click inside the graph area to view the respective abscissa `t` and ordinate.

Back to Content [page 482]

1.4.5.3.6.3 Monitoring APIs

Use the Cloud Connector monitoring APIs to include monitoring information in your own monitoring tool.

Context

You might want to integrate some monitoring information in the monitoring tool you use.

For this purpose, the Cloud Connector includes a collection of APIs that allow you to read various types of monitoring data.

> **i Note**
>
> This API set is designed particularly for monitoring the Cloud Connector via the SAP Solution Manager, see Configure Solution Management Integration [page 457].

Prerequisites

You must use *Basic Authentication* or *form field* authentication to read the monitoring data via API.

Users must be assigned to the roles `sccmonitoring` or `sccadmin`. The role `sccmonitoring` is restricted to managing the monitoring APIs.

> **i Note**
>
> The Health Check API does not require a specified user. Separate users are available through LDAP only.

URL Parameters

The API URLs adhere to the following pattern:

`https://<scchost>:<sccport>/xxx`

- Only HTTPS is supported
- `<scchost>:<sccport>` = cloud connector address
- `xxx`= path of the calling method

Available APIs

The following APIs are currently available.

- Health Check [page 488] (available as of version 2.6.0)
- Subaccount data [page 488] (as of 2.10.0)
- Connection data [page 490] (as of 2.10.0)
- Performance data [page 492] (as of 2.10.0)
- Top Time Consumers [page 494] (as of 2.11.0)
- Memory Status [page 496] (as of 2.13.0)
- Certificate Status [page 498] (as of 2.13.0)
- Usage Statistics [page 500] (as of 2.13.0)

Health Check (available as of version 2.6.0)

> **i Note**
>
> This API is relevant for the master instance as well as for the shadow instance.

Using the health check API, it is possible to recognize that the Cloud Connector is up and running. The purpose of this health check is only to verify that the Cloud Connector is not down. It does not check any internal state or tunnel connection states. Thus, it is a quick check that you can execute frequently:

URL	`https://<scc_host>:<scc_port>/exposed?action=ping`
Expected Return Code	`200`

Back to Available APIs [page 488]

Back to Context [page 487]

List of Subaccounts (available as of version 2.10.0)

> **i Note**
>
> This API is relevant for the master instance only.

Using this API, you can read the list of all subaccounts connected to the Cloud Connector and view detail information for each subaccount:

URL	**`https://<scchost>:<sccport>/api/monitoring/subaccounts`**
Input	None

Output	JSON document with list of detailed subaccount data, like:
	• `subaccounts`: array of subaccounts for which the data is provided
	  ○ `regionHost`: host of the region, in which the subaccount is residing
	  ○ `subaccount`: name of subaccount
	  ○ `locationID`: identifying the location of this Cloud Connector for a specific subaccount
	  ○ `tunnel`: array of connection tunnels used by the subaccount
	  ○ `state`: connected or disconnected
	  ○ `connectedSince`: connection start time
	  ○ `connections`: number of subaccount connections
	  ○ `applicationConnections`: array of connections to application instances
	  ○ `serviceChannels`: type and state of the service channels used (types: HANA database, Virtual Machine or RFC)
	  ○ `recoveryAccountState`: state and more details about the disaster recovery subaccount, if configured
	  ○ `isActive`: disaster recovery subaccount is working (true or false)
	• `version`: API version

Example:

```
{
  "subaccounts": [
    {
      "tunnel": {
        "state": "Connected",
        "connectedSince": "2017-05-11T15:51:14.915 +0200",
        "connections": 0,
        "applicationConnections": [],
        "serviceChannels": [
          {
            "type": "VirtualMachine",
            "state": "ConnectFailure",
            "details": "dexlab"
          }
        ]
      },
      "regionHost": "hana.ondemand.com",
      "subaccount": "a117",
      "locationID": ""
    },
    {
      "recoveryAccountState": {
        "isActive": false,
        "tunnel": {
          "state": "Connected",
          "connectedSince": "2017-05-12T06:58:26.613 +0200",
          "connections": 0,
          "applicationConnections": [],
          "serviceChannels": []
        },
        "regionHost": "sapbydesign.com",
        "subaccount": "dev",
        "locationID": ""
```


Back to Available APIs [page 488]

Back to Context [page 487]

List of Open Connections (available as of version 2.10.0)

> **i** Note
>
> This API is relevant for the master instance only.

The list of connections lets you view all back-end systems connected to the Cloud Connector and get detail information for each connection:

URL	**`https://<scchost>:<sccport>/api/monitoring/connections/backends`**
Input	None

Output	JSON document with list of all open connections and detailed information about back-end systems:
	• `subaccounts`: array of subaccounts for which the data is provided
	  ○ `backendConnections`: array of connections to a specified back-end system
	  ○ `virtualBackend`: virtual (external) back-end URL
	  ○ `internalBackend`: internal back-end URL
	  ○ `protocol`: type of protocol (RFC, HTTP, and so on)
	  ○ `idle`: number of idle connections
	  ○ `active`: number of active connections
	  ○ `state`: connected or disconnected
	• `version`: API version

Example:

```
{
  "subaccounts": [
    {
      "backendConnections": [],
      "regionHost": "hana.ondemand.com",
      "subaccount": "a117",
      "locationID": ""
    },
    {
      "backendConnections": [
        {
          "virtualBackend": "abapserver.hana.cloud:",
          "internalBackend": "sap.corp:",
          "protocol": "RFC",
          "idle": 1,
          "active": 0
        }
      ],
      "regionHost": "hana.ondemand.com",
      "subaccount": "dev",
      "locationID": ""
    },
    {
      "backendConnections": [],
      "regionHost": "hanatrial.ondemand.com",
      "subaccount": "trial",
      "locationID": ""
    }
  ],
  "version": 1
}
```


Back to Available APIs [page 488]

Back to Context [page 487]

Performance Monitor Data (available as of version 2.10.0)

> **i** Note
>
> This API is relevant for the master instance only.

Using this API, you can read the data provided by the Cloud Connector performance monitor:

URL	`https://<scchost>:<sccport>/api/monitoring/performance/backends`

Input	None

Output	JSON document with a list providing the Cloud Connector performance monitor data with detailed information about back-end performance:
	• `subaccounts`: array of subaccounts for which data is provided
	  ○ `VirtualHost`: host name of the back-end system
	  ○ `VirtualPort`: port of the back-end system
	  ○ `Protocol`: type of protocol (RFC, HTTP, and so on)
	  ○ `Buckets`: array of performance data related to back-end system
	  ○ `numberOfCalls`: number of calls performed between Cloud Connector and back-end system
	  ○ `miniumCallDurationsMs`: minimum duration of the executed calls in milliseconds
	  ○ `SinceTime`: start of performance measurement
	• `version`: API version

Example:

```
{
  "subaccounts": [
    {
      "backendPerformance": [],
      "sinceTime": "2017-05-11T15:48:42.084 +0200",
      "regionHost": "hana.ondemand.com",
      "subaccount": "a117",
      "locationID": ""
    },
    {
      "backendPerformance": [
        {
          "virtualHost": "abapserver.hana.cloud",
          "virtualPort": "2",
          "protocol": "RFC",
          "buckets": [
            {
              "numberOfCalls": 0,
              "minimumCallDurationMs": 0
            },
            {
              "numberOfCalls": 1,
              "minimumCallDurationMs": 10
            },
            {
              "numberOfCalls": 0,
              "minimumCallDurationMs": 20
            },
            {
              "numberOfCalls": 0,
              "minimumCallDurationMs": 30
            },
```


Back to Available APIs [page 488]

Back to Context [page 487]

Top Time Consumers (available as of version 2.11.0)

> **i** Note
>
> This API is relevant for the master instance only.

Using this API, you can read the data of top time consumers provided by the Cloud Connector performance monitor:

URL	`https://<scchost>:<sccport>/api/monitoring/performance/toptimeconsumers`
Input	None

Output	JSON document with list of top time consumer data:
	• `subaccounts`: array of subaccounts for which the data is provided
	  ○ `requests`: array of requests sent to a specified backend system, including performance data
	  ○ `protocol`: type of protocol (RFC, HTTP, and so on)
	  ○ `virtualBackend`: virtual (external) back-end URL
	  ○ `internalBackend`: internal back-end URL
	  ○ `resource`: name of the request resource
	  ○ `sentBytes`: number of sent bytes
	  ○ `receivedBytes`: number of received bytes
	  ○ `user`: name of the request user
	  ○ `totalTime`: total request time in milliseconds
	  ○ `externalTime`: in milliseconds
	  ○ `genSsoTime`: in milliseconds
	  ○ `openRemoteTime`: in milliseconds
	  ○ `validationSsoTime`: time for SSO validation in milliseconds
	  ○ `latencyTime`: latency in milliseconds
	• `version`: API version

Example:

```
{
  "subaccounts": [
    {
      "sinceTime": "2017-06-26T16:57:21.615 +0200",
      "requests": [
        {
          "startTime": "2017-06-26T16:57:25.426 +0200",
          "id": 1639964397,
          "protocol": "RFC",
          "virtualBackend": "abapserver.hana.cloud:sapgw42",
          "internalBackend": "ldciv9u:sapgw51",
          "resource": "RFCPING",
          "sentBytes": 307,
          "receivedBytes": 441,
          "user": "JCOUSER",
          "totalTime": 21,
          "externalTime": 6,
          "genSsoTime": 0,
          "openRemoteTime": 5,
          "validateSsoTime": 0,
          "latencyTime": 6
        },
        { … } // 15 items
      ],
      "regionHost": "int.sap.hana.ondemand.com",
      "subaccount": "d039407sapdev",
      "locationID": "location"
    }
  ],
  "version": 1
}
```


Back to Available APIs [page 488]

Back to Context [page 487]

Memory Status (available as of version 2.13.0)

> **i Note**
>
> This API is relevant for the master instance only.

This API provides a snapshot of the current memory status of the machine where the Cloud Connector is running:

URL	`https://<scchost>:<sccport>/api/monitoring/memory`
Input	None

Output	JSON document with memory data:
	• `physicalKB`: usage of the physical memory, split into four categories (all sizes in KB):
	  ○ `total`: total size of the physical memory
	  ○ `CloudConnector`: size of the physical memory used by the Cloud Connector
	  ○ `others`: size of the physical memory used by all other processes
	  ○ `free`: size of the free physical memory
	• `virtualKB` : usage of the virtual memory, split into four categories (all sizes in KB)
	  ○ `total`: total size of the virtual memory
	  ○ `CloudConnector`: size of the virtual memory used by the Cloud Connector
	  ○ `others`: size of the virtual memory used by all other processes
	  ○ `free`: size of the free virtual memory
	• `cloudConnectorHeapKB` : usage of the Java heap, split into three categories (all sizes in KB):
	  ○ `total`: total size of the Java heap
	  ○ `used`: size of the Java heap used by the Cloud Connector
	  ○ `free`: size of the free Java heap

Example:

```
{
  "physicalKB": {
    "total": 33380516,
    "CloudConnector": 633932,
    "others": 10960464,
    "free": 21786120
  },
  "virtualKB": {
    "total": 35477668,
    "CloudConnector": 1267504
    "others": 14174208,
    "free": 20035956
  },
  "cloudConnectorHeapKB": {
    "total": 983040,
    "used": 235457,
    "free": 747583
  }
}
```

Back to Available APIs [page 488]

Back to Context [page 487]

Certificate Status (available as of version 2.13.0)

> **i** Note
>
> This API is relevant for the master instance only.

Using this API, you can get an overview of the certificates currently employed by the Cloud Connector:

URL	`https://<scchost>:<sccport>/api/monitoring/certificates`
Input	None

Output	JSON document with data reflecting the current status of all certificates. There are three categories or lists, each of them containing zero or more certificates:
	
	• `expired`: list of all expired certificates
	• `expiring`: list of all certificates that will expire in less than N days, where N is the number of days specified in the alerting setup regarding certificates that are close to their expiration date.
	• `ok` : list of all certificates that continue to be valid for N days or more, where N is the number of days specified in the alerting setup regarding certificates that are close to their expiration date.
	
	A certificate in any of those lists is represented by a JSON object with the following properties:
	
	• `type`: type of the certificate which can be one of the following strings:
	  ○ `UI` (for the UI certificate)
	  ○ `System` (for the system certificate)
	  ○ `CA` (for the certificate used in connection with principal propagation/certification authority)
	  ○ `subaccount` (for subaccount certificates)
	• `validTo`: end date of the respective certificate's validty (as a long integer, i.e. a UTC timestamp)
	• `subjectDN`: subject DN of the respective certificate (included only for non-subaccount certificates)
	• `subaccountName`: name of the subaccount (only for subaccount certificates)
	• `subaccountRegion`: region or landscape host of the the subaccount (only for subaccount certificates)
	• `isDisasterRecoverySubaccount`: flag (Boolean value) that indicates that the subaccount is employed for disaster recovery. It can therefore be present only for subaccount certificates. Moreover, it is added only for disaster recovery subaccounts with its value set to true.

URL	**`https://<scchost>:<sccport>/api/monitoring/certificates/expired`**
Input	None
Output	JSON document (an array) holding the list of all expired certificates.
URL	**`https://<scchost>:<sccport>/api/monitoring/certificates/expiring`**
Input	None

Output	JSON document (an array) holding the list of all certificates that will expire in less than N days, where N is the number of days specified in the alerting setup regarding certificates that are close to their expiration date.
URL	`https://<scchost>:<sccport>/api/monitoring/certificates/ok`
Input	None
Output	JSON document (array) holding the list of all certificates that continue to be valid for N days or more, where N is the number of days specified in the alerting setup regarding certificates that are close to their expiration date.

Example:


```
{
  "expired": [
    {
      "type": "System",
      "subjectDN": "CN\u003dHugo, OU\u003dCSI, O\u003dSAP Trust Community, C\u003dDE",
      "validTo": 1458290342000
    }
  ],
  "expiring": [],
  "ok": [
    {
      "type": "UI",
      "subjectDN": "CN\u003dSCC, OU\u003dConnectivity, O\u003dSAP SE, C\u003dDE",
      "validTo": 1791826434000
    },
    {
      "type": "subaccount",
      "subaccountName": "d036325trial",
      "subaccountRegion": "hanatrial.ondemand.com",
      "validTo": 1613132865000
    }
  ]
}
```


Back to Available APIs [page 488]

Back to Context [page 487]

Usage Statistics (available as of version 2.13.0)

> **i Note**
>
> This API is relevant for the master instance only.

This API provides usage statistics regarding the systems and resources available in the Cloud Connector:

URL	`https://<scchost>:<sccport>/api/monitoring/performance/toptimeconsumers`
Input	None

Output

JSON document (a JSON object) with usage data. There is one property:

- `subaccounts`: list of subaccounts (a JSON array)

Each subaccount in the subaccounts array is represented through an object with the following properties:

- `subaccountName`: name of the subaccount
- `subaccountRegion`: region host for the subaccount
- `sinceTime`: time at which collecting statistics started for this account (a UTC timestamp)
- `usageStatistics`: usage statistics, given as an array.

Each element of the usageStatistics array corresponds to one system (virtual host) and is represented through an object with the following properties:

- `virtualHost`: name of the virtual host
- `virtualPort`: port of the virtual host
- `protocol`: protocol this virtual host can process.

> **i Note**
>
> Currently, statistical data is collected only for protocols HTTP, HTTPS, RFC, and RFC SNC, and hence systems relying on other protocols are not listed here.

- `bytesReceived`: total number of bytes that were received through a call or request
- `bytesSent`: total number of bytes sent back as a response
- `calls`: total number of calls or requests
- `mostRecentAccess`: time of the most recent access (call or request) given as a UTC timestamp.

> **i Note**
>
> This property is only available if there has been at least one call or request.

- `resources`: usage statistics per resource, given as an array (i.e. the distribution of bytes received, sent, as well as number of calls/requests, across the resources of the respective virtual host).

Each element of the resources array holds the usage statistics of a resource, represented through an object with the following properties:

- `resourceName`: name of the resource (a URL path or the name of a remote function)

- `enabled`: Boolean flag that indicates whether the resource is currently active (true) or suspended (false).
- `bytesReceived`: total number of bytes that were received through a call or request and were handled by this resource.
- `bytesSent`: total number of bytes sent back as a response in the context of this resource.
- `calls`: total number of calls or requests handled by this resource
- `mostRecentAccess`: time of the most recent access (i.e., call or request) given as a UTC timestamp.

  > **Note**
  >
  > This property is only available if at least one call or request was handled by this resource.

> **Note**
>
> Usage statistics are collected at runtime and stored in memory. They are lost when stopping or restarting the Cloud Connector.

Example:

```
{
    "subaccounts": [
        {
            "subaccountName": "lannister",
            "subaccountRegion": "westeros.com",
            "sinceTime": 1596810677689,
            "usageStatistics": [
                {
                    "virtualHost": "iron.islands",
                    "virtualPort": "1234",
                    "protocol": "RFC",
                    "bytesReceived": 0,
                    "bytesSent": 0,
                    "calls": 0,
                    "resources": [
                        {
                            "resourceName": "TEST_FCT_1",
                            "enabled": true,
                            "bytesReceived": 0,
                            "bytesSent": 0,
                            "calls": 0
                        },
                        {
                            "resourceName": "TEST_FCT_2",
                            "enabled": true,
                            "bytesReceived": 0,
                            "bytesSent": 0,
                            "calls": 0
                        }
                    ]
                },
                {
                    "virtualHost": "the.north",
                    "virtualPort": "8080",
                    "protocol": "HTTP",
                    "bytesReceived": 1482,
                    "bytesSent": 705,
                    "calls": 3,
                    "mostRecentAccess": 1596810970078,
                    "resources": [
                        {
                            "resourceName": "/invoke/giants",
                            "enabled": true,
                            "bytesReceived": 0,
                            "bytesSent": 0,
                            "calls": 0
                        },
                        {
                            "resourceName": "/invoke/whiteWalkers",
                            "enabled": true,
                            "bytesReceived": 0,
                            "bytesSent": 0,
                            "calls": 0
                        },
                        {
                            "resourceName": "/invoke/nightWatch",
                            "enabled": true,
                            "bytesReceived": 1482,
                            "bytesSent": 705,
                            "calls": 3,
                            "mostRecentAccess": 1596810970078
                        }
                    ]
                }
            ]
        }
```

Back to Available APIs [page 488]

Back to Context [page 487]

1.4.5.3.7 Alerting

Configure the Cloud Connector to send e-mail messages when situations occur that may prevent it from operating correctly.

To configure alert e-mails, choose *Alerting* from the top-left navigation menu.

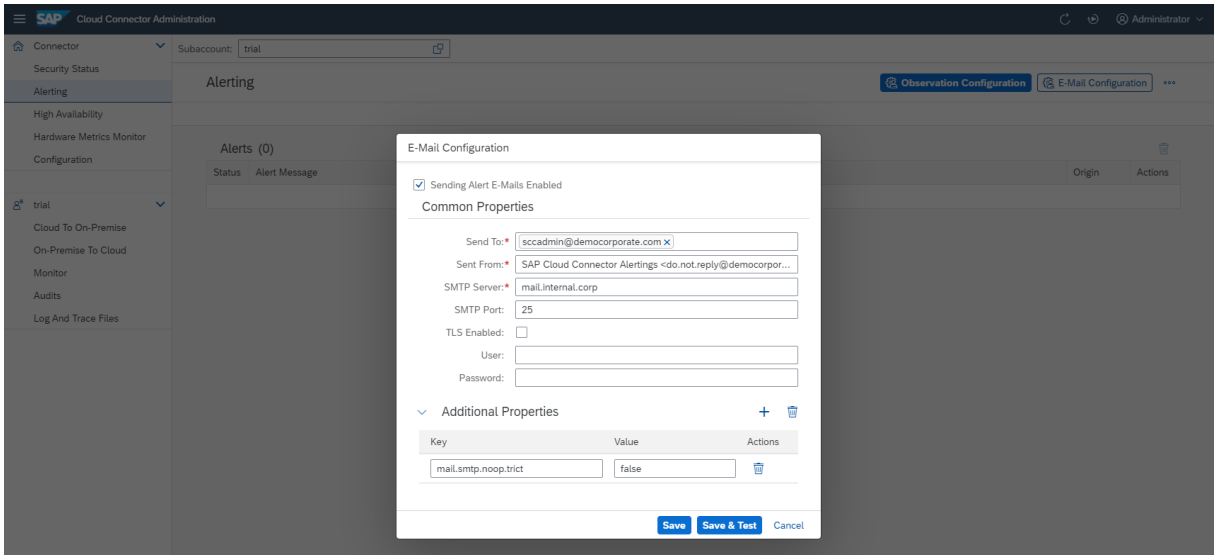
You must specify the receivers of the alert e-mails (*E-mail Configuration*) as well as the Cloud Connector resources and components that you want to monitor (*Observation Configuration*). The corresponding *Alert Messages* are also shown in the Cloud Connector administration UI.

E-mail Configuration

1. Select *E-mail Configuration* to specify the list of em-ail addresses to which alerts should be sent (*Send To*).

   > **i Note**
   >
   > The addresses you enter here can use either of the following formats: john.doe@company.com or John Doe <j.doe@company.com>.

2. Enter the sender's e-mail address (`<Sent From>`).
3. In `<SMTP Server>` provide the host of the mail server.
4. You can specify an `<SMTP port>`, if the server is not using the default ports. For details, contact your e-mail administrator or provider.
5. Mark the *TLS Enabled* check box if you want to establish a TLS-encrypted connection.
6. If the SMTP server requires authentication, provide `<User>` and `<Password>`.
7. In the *Additional Properties* section you can provide any property supported by the Java Mail library . All specified properties will be passed to the SMTP client.
8. Select *Save* to change the current configuration.



> ### Note
>
> Connections to an SMTP server over SSL/TLS can cause SSL errors if the SMTP server uses an "untrusted" certificate. If you cannot use a trusted certificate, you must import the public part of the issuer certificate to the JDK's trust storage.
>
> Usually, the trust storage is done in the file *cacerts* in the Java directory (`jre/lib/security/cacerts`). For import, you can use the *keytool* utility:
>
> ```
> keytool -import -storepass changeit -file <certificate used by SMTP server> -
> keystore cacerts -alias <for example, SMTP_xyz>
> ```
>
> For more information, see https://docs.oracle.com/cd/E19830-01/819-4712/ablqw/index.html .

Observation Configuration

Once you've entered the e-mail addresses to receive alerts, the next step is to identify the resources and components of the Cloud Connector: E-mail messages are sent when any of the chosen components or resources have malfunctioned or are in a critical state.

> ### Note
>
> The Cloud Connector does not dispatch the same alert repeatedly. As soon as an issue has been resolved, an informational alert is generated, sent, and listed in *Alert Messages* (see section below).

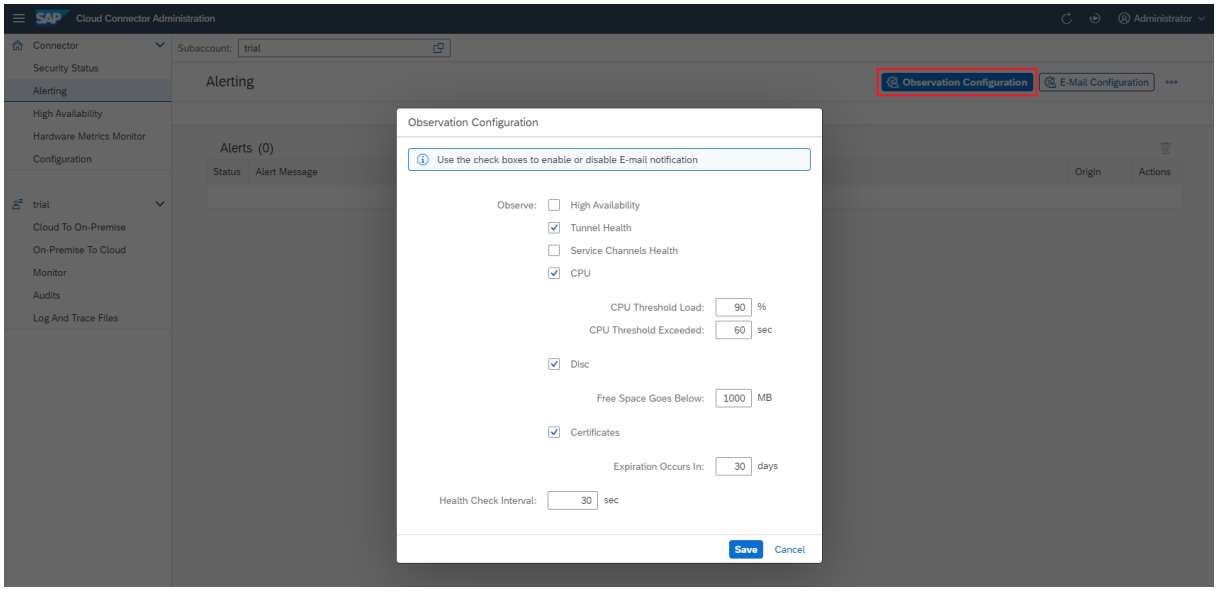
1. Select *Observation Configuration* from the top-right corner of the window.
2. Select the components or resources you want to monitor.
   - *High Availability* alerts can occur in the context of an active high availability setup, meaning a shadow system is connected.

- *Tunnel Health* and *Service Channels Health* refer to the state of the respective connections. Whenever such a connection is lost, an alert is triggered.

  > **i Note**
  >
  > These alerts are only triggered in case of an error or exception, but not upon intentional disconnect action.

- An excessively high *CPU* load over an extended period of time adversely affects performance and may be an indicator of serious issues that jeopardize the operability of the Cloud Connector. The CPU load is monitored and an alert is triggered whenever the CPU load exceeds and continues to exceed a given threshold percentage (the default is 90%) for more than a given period of time (the default is 60 seconds).
- Although the Cloud Connector does not require nor consume large amounts of *Disk* space, running out of it is a circumstance that you should avoid. We recommend that you configure an alert to be sent if the disk space falls below a critical value (the default is 10 megabytes).
- The Cloud Connector configuration contains various *Certificates*. Whenever one of those expires, scenarios might no longer work as expected so it's important to get notified about the expiration (the default is 30 days).

3. (Optional) Change the *Health Check Interval* (the default is 30 seconds).
4. Select *Save* to change the current configuration.



Alert Messages

The Cloud Connector shows alert messages also on screen, in *Alerting* > *Alert Messages*.

You can remove alerts using *Delete* or *Delete All*. If you delete active (unresolved) alerts, they reappear in the list after the next health check interval.

1.4.5.3.8 Audit Logging

Audit log data can alert Cloud Connector administrators to unusual or suspicious network and system behavior.

Additionally, the audit log data can provide auditors with information required to validate security policy enforcement and proper segregation of duties. IT staff can use the audit log data for root-cause analysis following a security incident.

The Cloud Connector includes an auditor tool for viewing and managing audit log information about access between the cloud and the Cloud Connector, as well as for tracking of configuration changes done in the Cloud Connector. The written audit log files are digitally signed by the Cloud Connector so that their integrity can be checked, see Manage Audit Logs [page 508].

> **i Note**
>
> We recommend that you permanently switch on Cloud Connector audit logging in productive scenarios.
>
> - Under normal circumstances, set the logging level to **`Security`** (the default configuration value).
> - If legal requirements or company policies dictate it, set the logging level to **`All`**. This lets you use the log files to, for example, detect attacks of a malicious cloud application that tries to access on-premise services without permission, or in a forensic analysis of a security incident.

We also recommend that you regularly copy the audit log files of the Cloud Connector to an external persistent storage according to your local regulations. The audit log files can be found in the Cloud Connector root directory `/log/audit/<subaccount-name>/audit-log_<timestamp>.csv`.

1.4.5.3.8.1 Manage Audit Logs

Configure audit log settings and verify the integrity of audit logs.

Configure Audit Logs in the Cloud Connector

Choose *Audit* from your subaccount menu and go to *Settings* to specify the type of audit events the Cloud Connector should log at runtime. You can currently select between the following *Audit Levels* (for either `<subaccount>` and `<cross-subaccount>` scope):

- **Security**: Default value. The Cloud Connector writes an audit entry (`Access Denied`) for each request that was blocked. It also writes audit entries, whenever an administrator changes one of the critical configuration settings, such as exposed back-end systems, allowed resources, and so on.
- **All**: The Cloud Connector writes one audit entry for each received request, regardless of whether it was allowed to pass or not (`Access Allowed` and `Access Denied`). It also writes audit entries that are relevant to the **Security** mode.
- **Off**: No audit entries are written.

> **i Note**
>
> We recommend that you don't log all events unless you are required to do so by legal requirements or company policies. Generally, logging security events only is sufficient.

To enable automatic cleanup of audit log files, choose a period (14 to 365 days) from the list in the field `<Automatic Cleanup>`.

Audit entries for configuration changes are written for the following different categories:

- `Account`: A subaccount configuration was changed.
- `RecoveryAccount`: A disaster recovery subaccount configuration was changed.
- `Configuration`: A new subaccount was added or a disaster recovery switch happened.
- `BackendMapping`: Changes to the virtual to internal system mappings.
- `AllowedResource`: In a virtual system, changes in the accessible resources.
- `DomainMapping`: Changes to the domain mappings.
- `ServiceChannelConfiguration`: The configuration of a service channel was changed.
- `SCCPassword`: The Cloud Connector administration password was changed.
- `SCCUser`: The Cloud Connector administration user was changed.
- `LDAPConfiguration`: Changes to the LDAP settings.
- `EmailConfiguration`: The Email settings for alerts were changed.
- `AlertConfiguration`: The observation settings for alerts were changed.
- `ScimConfiguration`: Something changed in the settings for the cloud user store.
- `KerberosConfiguration`: The Kerberos configuration was changed.
- `SNCSettings`: SNC settings of Cloud Connector were changed.
- `ProxySettings`: The proxy settings were changed.
- `SystemCertificate`: The system certificate was changed.
- `PpcaCertificate`: The CA certificate was changed.
- `PrincipalPropagationConfiguration`: The principal propagation settings were changed.
- `TrustSynchronization`: The trust configuration for principal propagation was synchronized.
- `IdentityProviderTrust`: The trust configuration for a specific identity provider was changed.
- `ApplicationTrust`: The trust configuration to applications was changed.
- `TrustedBackendCertificate`: The trust store certificate was added or removed.
- `SccCustomRoles`: Custom role name settings were changed.
- `BackendAuthority`: RFC-specific user and client settings were adjusted.
- `AdvancedConnectivity`: Advanced connectivity configuration was changed.
- `AdvancedJVM`: Advanced JVM configuration was changed.
- `ApplicationConfiguration`: Application-specific connection configuration was changed.
- `PayloadTrace`: Payload trace (traffic data) was activated/deactivated.
- `CPICTrace`: The CPIC trace level was changed.
- `AuditLogLevel`: The subaccount-specific audit log level was changed.
- `CrossAuditLogLevel`: The cross-subaccount audit log level was changed.
- `AuditLogCleanup`: The audit log cleanup setting was changed.

In the *Audit Viewer* section, you can first define filter criteria, then display the selected audit entries.

- In the *Audit Type* field, you can select whether to view the audit entries for the following:
  - Any entries
  - Only denied requests
  - Only allowed requests
  - Service started
  - Service stopped
  - Cloud Connector changes
  - High Availability
  - Principal Propagation
- In the *Pattern* field, you can specify a certain string that the detail text of each selected audit entry must contain. The detail text contains information about the user name, requested resource/URL, and the virtual `<host>:<port>`. Wildcards are currently not supported. Use this feature to do the following:
  - Filter the audit log for all requests that a particular HTTP user has made during a certain time frame.
  - Identify all users who attempted to request a particular URL.
  - Identify all requests to a particular back-end system.
  - Determine whether a user has changed a certain Cloud Connector configuration. For example, a search for string **`BackendMapping`** returns all `add-`, `delete-` or `modify-` operations on the *Mapping Virtual To Internal System* page.
- The *Time Range* settings specify the time frame for which you want to display the audit events.

These filter criteria are combined with a logical `AND` so that all audit entries that match these criteria are shown. If you have modified one of the criteria, select *Refresh* to display the updated selection of audit events that match the new criteria.

> **i Note**
>
> To prevent a single person from being able to both change the audit log level, and delete audit logs, we recommend that the operating system administrator and the SAP BTP administrator are different persons. We also suggest that you turn on the audit log at the operating system level for file operations.

The ◇ *Check* button checks all files that are filtered by the specified date range.

Verify the Integrity of Audit Logs

To check the integrity of the audit logs, go to `<scc_installation>/auditor`. This directory contains an executable `go` script file (respectively, `go.cmd` on Microsoft Windows and `go.sh` on other operating systems).

If you start the `go` file without specifying parameters from `<scc_installation>/auditor`, all available audit logs for the current Cloud Connector installation are verified.

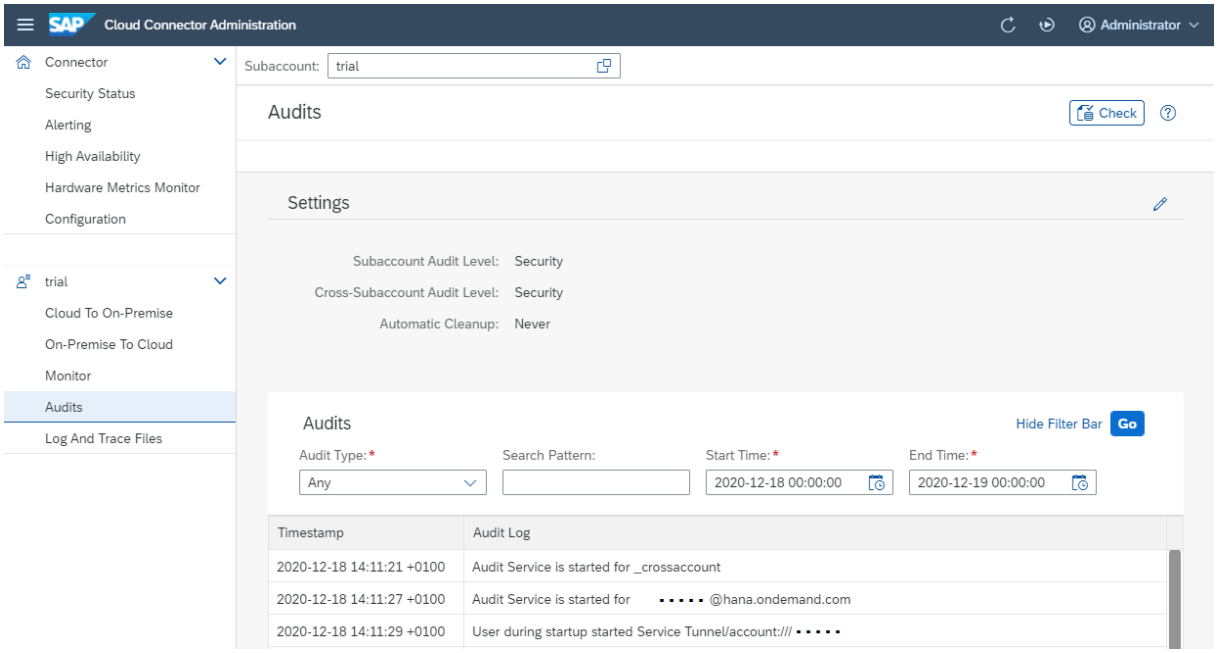
The auditor tool is a Java application, and therefore requires a Java runtime, specified in JAVA_HOME, to execute:

- For Microsoft Windows OS, set *JAVA_HOME=<path-to-java-installation>*
- For Linux OS and Mac OS X, export: *JAVA_HOME=<path-to-java-installation>*

Alternatively, to execute Java, you can include the Java `bin` directory in the `PATH` variable.

Example

In the following example, the *Audit Viewer* displays `Any` audit entries, at **`Security`** level, for the time frame between **December 18 2020, 00:00:00** and **December 19, 00:00:00**:



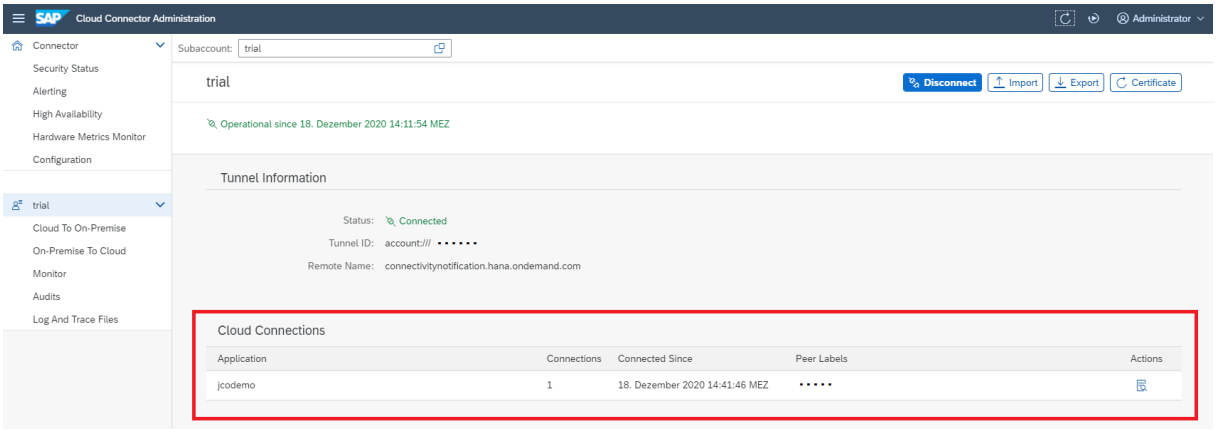
1.4.5.3.9 Troubleshooting

To troubleshoot connection problems, monitor the state of your open tunnel connections in the Cloud Connector, and view different types of logs and traces.

> **i** Note
>
> For information about a specific problem or an error you have encountered, see Connectivity Support [page 544].

Monitoring

To view a list of all currently connected applications, choose your *Subaccount* from the left menu and go to section *Cloud Connections*:



The provided information includes:

- *Application name*: The name of the application, as also shown in the cockpit, for your subaccount
- *Connections*: The number of currently existing connections to the application
- *Connected Since*: The earliest start time of a connection to this application
- *Peer Labels*: The name of the application processes, as also shown for this application in the cockpit, for your subaccount

Log and Trace Settings

The *Log and Trace Files* page includes some files for troubleshooting that are intended primarily for SAP Support. These files include information about both internal Cloud Connector operations and details about the communication between the local and the remote (SAP BTP) tunnel endpoint.

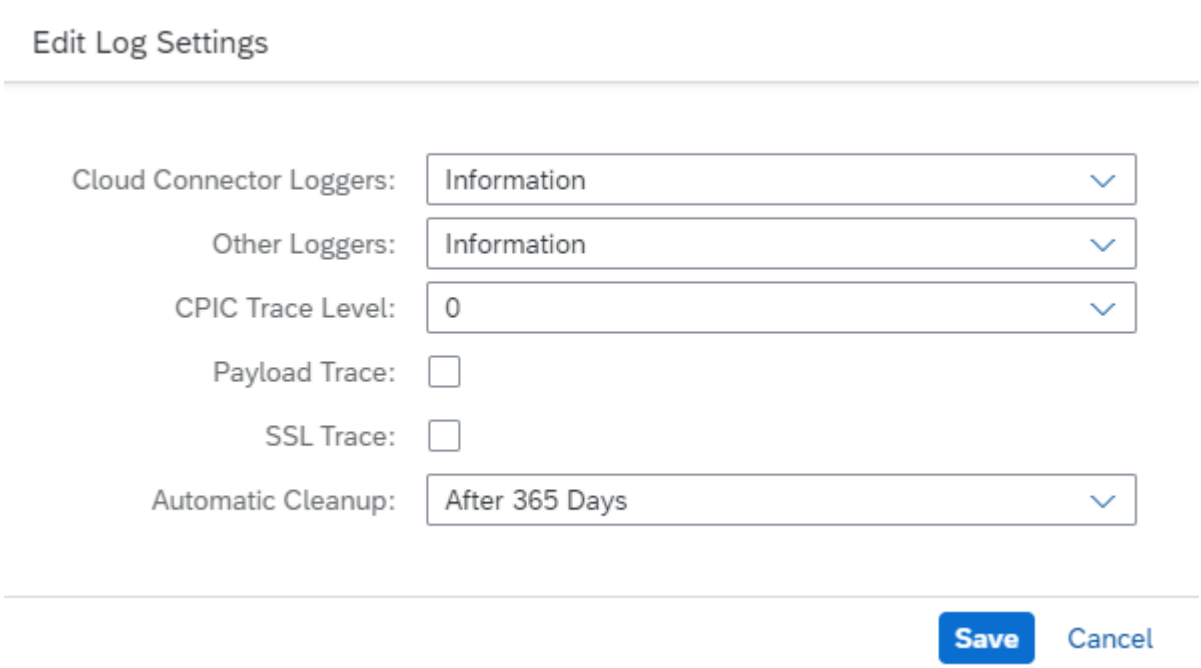
If you encounter problems that seem to be caused by some trouble in the communication between your cloud application and the on-premise system, choose *Log and Trace Files* from your *Subaccount* menu, go to section *Settings*, and activate the respective traces by selecting the *Edit* button:

- *Cloud Connector Loggers* adjusts the levels for Java loggers directly related to Cloud Connector functionality.
- *Other Loggers* adjusts the log level for all other Java loggers available at the runtime. Change this level only when requested to do so by SAP support. When set to a level higher than `Information`, it generates a large number of trace entries.
- *CPIC Trace Level* allows you to set the level between 0 and 3 and provides traces for the CPIC-based RFC communication with ABAP systems.
- When the *Payload Trace* is activated for a subaccount, all the HTTP and RFC traffic crossing the tunnel for that subaccount going through this Cloud Connector, is traced in files with names `traffic_trace_<subaccount id>_on_<regionhost>.trc`.

> **i Note**
>
> Use payload and CPIC tracing at Level 3 carefully and only when requested to do so for support reasons. The trace may write sensitive information (such as payload data of HTTP/RFC requests and responses) to the trace files, and thus, present a potential security risk. As of version 2.2, the Cloud Connector supports an implementation of a "four-eyes principle" for activating the trace levels that

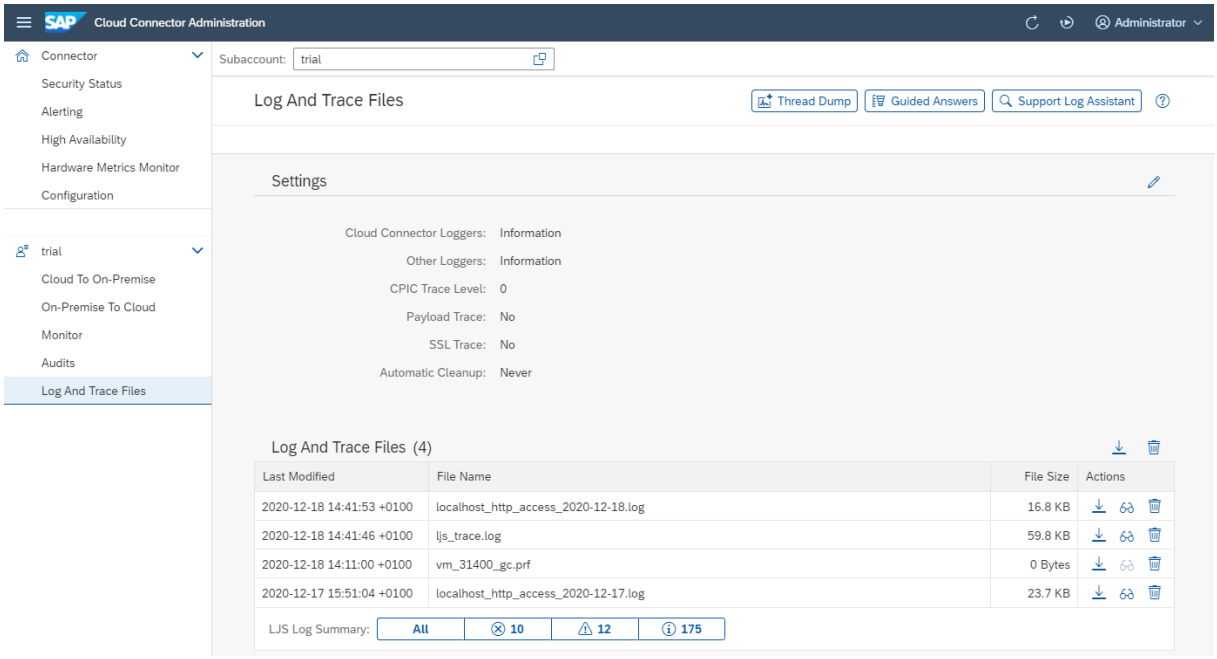
> dump the network traffic into a trace file. This principle requires two users to activate a trace level that records traffic data. See Secure the Activation of Traffic Traces [page 477].

- *SSL Trace*: When the SSL trace is activated, the `ljs_trace.log` file includes information for SSL-protected communication. To activate a change of this setting, a restart is required. Activate this trace only when requested by SAP support. It has a high impact on performance as it produces a large amount of traces.
- *Automatic Cleanup* lets you remove old trace files that have not been changed for a period of time exceeding the configured interval. You can choose from a list of predefined periods. The default is `Never`.



Log and Trace Files

View all existing trace files and delete the ones that are no longer needed.



To prevent your browser from being overloaded when multiple large files are loaded simultaneously, the Cloud Connector loads only one page into memory. Use the page buttons to move through the pages.

Use the *Download*/*Download All* icons to create a ZIP archive containing one trace file or all trace files. Download it to your local file system for convenient analysis.

> **i Note**
>
> If you want to download more than one file, but not all, select the respective rows of the table and choose *Download All*.

When running the Cloud Connector with SAP JVM, it is possible to trigger the creation of a thread dump by pressing the *Thread Dump* button, which will be written to the JVM trace file *vm_$PID_trace.log* . You will be requested by SAP support to create one if it is expected to help during incident analysis.

> **i Note**
>
> From the UI, you can't delete trace files that are currently in use. You can delete them from the Linux OS command line; however, we recommend that you do not use this option to avoid inconsistencies in the internal trace management of the Cloud Connector.

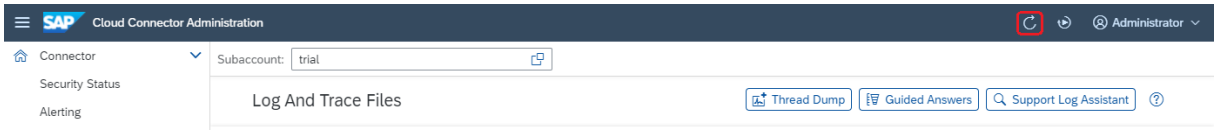
Two buttons may be helpful to solve issues on your own:

- *Guided Answers*: A new tab or window opens, showing the Cloud Connector section in Guided Answers. It helps you identify many issues that are classified through hierarchical topics. Once you found a matching issue, a solution is provided either directly, or by references to SAP Help Portal, Knowledge Base Articles (KBAs), and SAP notes.
- *Support Log Assistant*: Opens the support log assistant. There, you can upload Cloud Connector log files and have them analyzed. After triggering the scan, the tool lists all issues for which a solution can be identified.

> **Note**
>
> The support log assistant analyzes the complete log. Therefore, also older issues may be found that are no longer relevant.

Once a problem has been identified, you should turn off the trace again by editing the trace and log settings accordingly to not flood the files with unnecessary entries.

Use the *Refresh* button to update the information that appears. For example, you can use this button because more trace files might have been written since you last updated the display.



Error Analysis and Support: Which Logs are Relevant?

If you contact SAP support for help, please always attach the appropriate log files and provide the timestamp or period, when the reported issue was observed. Depending on the situation, different logs may help to find the root cause.

Some typical settings to get the required data are listed below:

- `<Cloud Connector Loggers>` provide **details related to connections to SAP BTP and to backend systems as well as master-shadow communication in case of a high availability setup**. However, it does not contain any payload data. This kind of trace is written into `ljs_trace.log`, which is the most relevant log for the Cloud Connector.
- `<Other Loggers>` provide **details related to the tomcat runtime**, in which the Cloud Connector is running. The traces are written into `ljs_trace.log` as well, but they are needed only in very special support situations. If you don't need these traces, leave the level on `Information` or even lower.
- **Payload data** are written into the traffic trace file for HTTP or RFC requests if the payload trace is activated, or into the CPI-C trace file for RFC requests, if the CPI-C trace is set to level 3.
- `<TLS trace>` is helpful to **analyze TLS handshake failures** from Cloud Connector to Cloud or from Cloud Connector to backend. It should be turned off again as soon as the issue has been reproduced and recorded in the traces.
- Setting the audit log on level `ALL` for `<Subaccount Audit Level>` is the easiest way to **check if a request reached the the Cloud Connector and if it is being processed**.

Related Information

Getting Support

1.4.5.3.10 Process Guidelines for Hybrid Scenarios

A hybrid scenario is one, in which applications running on SAP BTP require access to on-premise systems. Define and document your scenario to get an overview of the required process steps.

Tasks

Document the Landscape of a Hybrid Solution [page 516]

Document Administrator Roles [page 517]

Document Communication Channels [page 517]

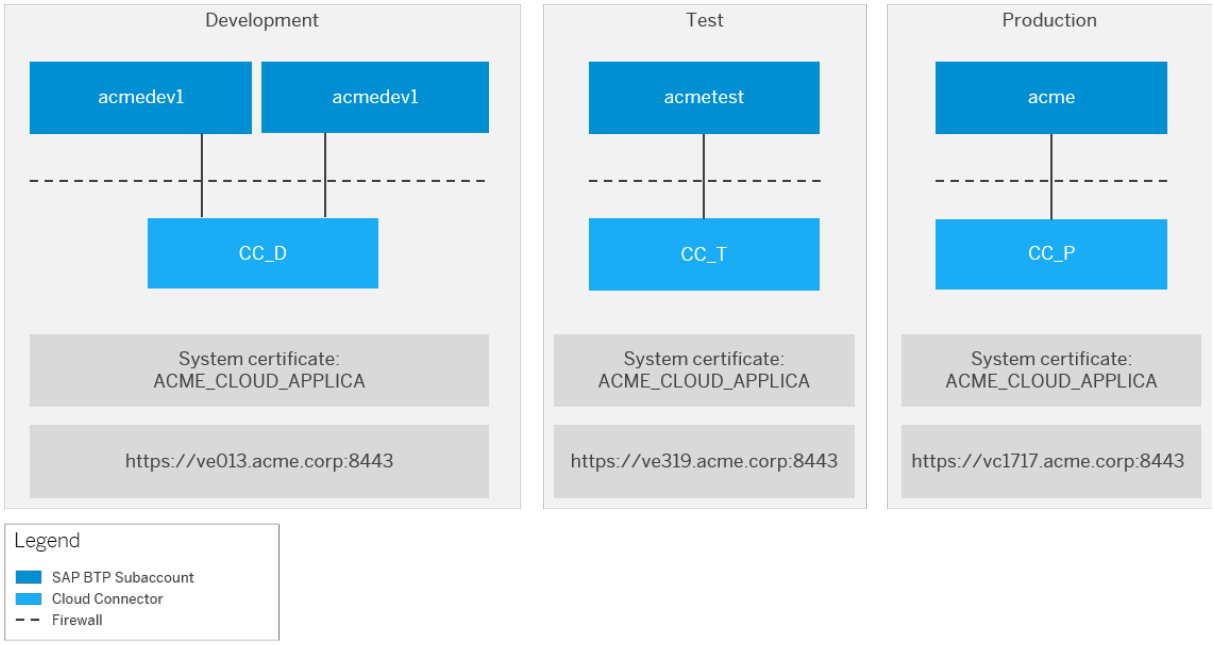
Define Project and Development Guidelines [page 518]

Define How to Set a Cloud Application Live [page 518]

Document the Landscape of a Hybrid Solution

To gain an overview of the cloud and on-premise landscape that is relevant for your hybrid scenario, we recommend that you diagrammatically document your cloud subaccounts, their connected Cloud Connectors and any on-premise back-end systems. Include the subaccount names, the purpose of the subaccounts (dev, test, prod), information about the Cloud Connector machines (host, domains), the URLs of the Cloud Connectors in the landscape overview document, and any other details you might find useful to include.

An example of landscape overview documentation could look like this:



Back to Tasks [page 516]

Document Administrator Roles

Document the users who have administrator access to the cloud subaccounts, to the Cloud Connector operating system, and to the Cloud Connector administration UI.

Such an administrator role documentation could look like following sample table:

Resource	john@acme.com	marry@acme.com	pete@acme.com	greg@acme.com
Cloud Subaccount (CA) Dev1	X			
CA Dev2		X		
CA Test			X	X
CA Prod				X
Cloud Connector Dev1 + Dev2	X	X		
Cloud Connector Test			X	X
Cloud Connector Prod				X
Cloud Connector Dev1 + Dev2 file system			X	X
Cloud Connector Test file system				X
Cloud Connector Prod file system				

Back to Tasks [page 516]

Document Communication Channels

Create and document separate email distribution lists for both the cloud subaccount administrators and the Cloud Connector administrators.

An example of the documented communication channels could look like this:

Landscape	Distribution List
Cloud Subaccount Administrators	DL ACME HCP Subaccount Admins
Cloud Connector Administrators	DL ACME Cloud Connector Admins

Back to Tasks [page 516]

Define Project and Development Guidelines

Define and document mandatory project and development guidelines for your SAP BTP projects. An example of such a guideline could be similar to the following.

Every SAP BTP project in this organization requires the following:

- Use Maven, Nexus, Git-&-Gerrit for the application development.
- Align with accountable manager in projects (including the names).
- Align with accountable security officer in projects (including the names).
- For externally developed source code, an official handover to the organization.
- Fulfill connection restrictions in a three-system landscape, that is, use a staged landscape for dev, test and prod, and, for example, the dev landscape connects only to dev systems, and so on.
- Productive subaccounts cannot use the same Cloud Connector as a dev or test subaccount.

Back to Tasks [page 516]

Define How to Set a Cloud Application Live

Define and document how to set a cloud application live and how to configure needed connectivity for such an application.

For example, the following processes could be seen as relevant and should be defined and document in more detail:

1. Transferring application to production: Steps for transferring an application to the productive status on the SAP BTP.
2. Application connectivity: The steps for adding a connectivity destination to a deployed application for connections to other resources in the test or productive landscape.
3. Cloud Connector Connectivity: Steps for adding an on-premise resource to the Cloud Connector in the test or productive landscapes to make it available for the connected cloud subaccounts.
4. On-premise system connectivity: The steps for setting up a trusted relationship between an on-premise system and the Cloud Connector, and to configure user authentication and authorization in the on-premise system in the test or productive landscapes.
5. Application authorization: The steps for requesting and assigning an authorization that is available inside the SAP BTP application to a user in the test or productive landscapes.
6. Administrator permissions: Steps for requesting and assigning the administrator permissions in a cloud subaccount to a user in the test or productive landscape.

Back to Tasks [page 516]

1.4.5.3.11 Configuring Backup

Find an overview of backup procedures for the Cloud Connector.

Configuration Backup [page 460]	Backup and restore your Cloud Connector configuration via the administration UI.
Backup [page 398]	Manage the Cloud Connector's configuration backup via REST API.
Backup And Restore Configuration [page 438]	Example: Backup and restore the Cloud Connector configuration via REST API.

1.4.5.4 Security

Learn how Cloud Connector features help you manage security.

Features

Security is a crucial concern for any cloud-based solution. It has a major impact on the business decision of enterprises whether to make use of such solutions. SAP BTP is a platform-as-a-service offering designed to run business-critical applications and processes for enterprises, with security considered on all levels of the on-demand platform:

Level	Features
Application Layer [page 520]	• Frontend security • Security standard-based application development
Service Layer [page 521]	• Identity and access management • Data protection • Regulatory compliance management
Cloud Infrastructure Layer [page 526]	• Network infrastructure and communication • Sandboxing • Intrusion detection and prevention
Physical and Environmental Layer [page 526]	• Strict physical access control • High availability and disaster recovery capabilities

The Cloud Connector enables integration of cloud applications with services and systems running in customer networks, and supports database connections from the customer network to SAP HANA databases running on

SAP BTP. As these are security-sensitive topics, this section gives an overview on how the Cloud Connector helps maintain security standards for the mentioned scenarios.

Target Audience

The content of this section concerns:

- System and IT administrators
- Technology consultants
- Solution architects

Related Information

Security Guidelines [page 526]

1.4.5.4.1 Application Layer

On application level, the main tasks to ensure secure Cloud Connector operations are to provide appropriate frontend security (for example, validation of entries) and a secure application development.

Product Security Standard

Basically, you should follow the rules given in the product security standard, for example, protection against cross-site scripting (XSS) and cross-site request forgery (XSRF).

Scope and Design

The scope and design of security measures on application level strongly depend on the specific needs of your application.

1.4.5.4.2 Service Layer

You can use SAP BTP Connectivity to securely integrate cloud applications with systems running in isolated customer networks.

Overview

After installing the Cloud Connector as integration agent in your on-premise network, you can use it to establish a persistent TLS tunnel to SAP BTP subaccounts.

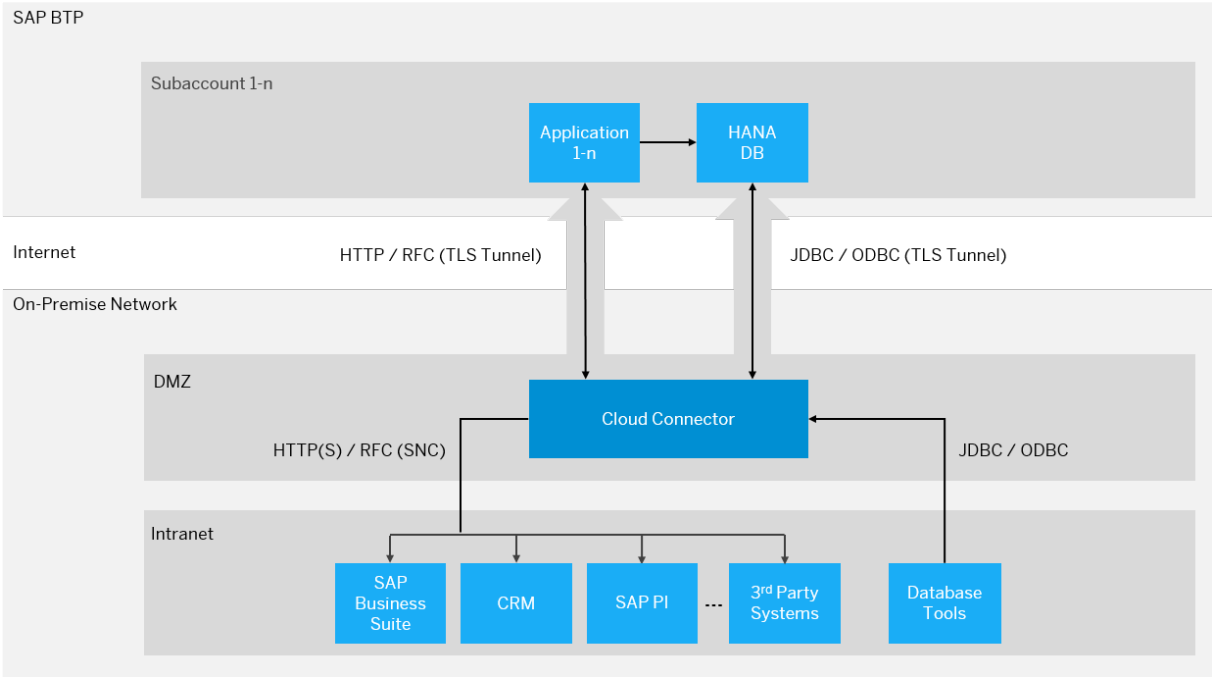
To establish this tunnel, the Cloud Connector administrator must authenticate himself or herself against the related SAP BTP subaccount of which he or she must be a member. Once estabilshed, the tunnel can be used by applications of the connected subaccount to remotely call systems in your network.

Architecture

The figure below shows a system landscape in which the Cloud Connector is used for secure connectivity between SAP BTP applications and on-premise systems.

- A single Cloud Connector instance can connect to multiple SAP BTP subaccounts, each connection requiring separate authentication and defining an own set of configuration.
- You can connect an arbitrary number of SAP and non-SAP systems to a single Cloud Connector instance.
- The on-premise system does not need to be touched when used with the Cloud Connector, unless you configure trust between the Cloud Connector and your on-premise system. A trust configuration is required, for example, for principal propagation (single sign-on), see Configuring Principal Propagation [page 301].
- You can operate the Cloud Connector in a high availability mode. To achieve this, you must install a second (redundant) Cloud Connector (shadow instance), which takes over from the master instance in case of a downtime.
- The Cloud Connector also supports the communication direction from the on-premise network to the SAP BTP subaccount, using a database tunnel that lets you connect common ODBC/JDBC database tools to SAP HANA as well as other available databases in SAP BTP.

.



Related Information

Network Zones [page 522]
Inbound Connectivity [page 523]
Outbound Connectivity [page 524]
Audit Log [page 525]

1.4.5.4.2.1 Network Zones

Choosing a network zone for the Cloud Connector installation.

A company network is usually divided into multiple network zones according to the security level of the contained systems. The DMZ network zone contains and exposes the external-facing services of an organization to an untrusted network, typically the Internet. Besides this, there can be one or multiple other network zones which contain the components and services provided in the company's intranet.

You can set up the Cloud Connector either in the DMZ or in an inner network zone. Technical prerequisites for the Cloud Connector to work properly are:

- The Cloud Connector must have access to the SAP BTP landscape host, either directly or via HTTPS proxy (see also: Prerequisites [page 239]).
- The Cloud Connector must have direct access to the internal systems it shall provide access to. I.e. there must be transparent connectivity between the Cloud Connector and the internal system.

It's a company's decision, whether the Cloud Connector is set up in the DMZ and operated centrally by an IT department or set up in the intranet and operated by the line of business.

Related Information

Network Zones [page 250]

1.4.5.4.2.2 Inbound Connectivity

For inbound connections into the on-premise network, the Cloud Connector acts as a reverse invoke proxy between SAP BTP and the internal systems.

Exposing Resources

Once installed, none of the internal systems are accessible by default through the Cloud Connector: you must configure explicitly each system and each service and resource on every system to be exposed to SAP BTP in the Cloud Connector.

You can also specify a virtual host name and port for a configured on-premise system, which is then used in the cloud. Doing this, you can avoid that information on physical hosts is exposed to the cloud.

TLS Tunnel

The TLS (Transport Layer Security) tunnel is established from the Cloud Connector to SAP BTP via a so-called **reverse invoke** approach. This lets an administrator have full control of the tunnel, since it can't be established from the cloud or from somewhere else outside the company network. The Cloud Connector administrator is the one who decides when the tunnel is established or closed.

The tunnel itself is using TLS with strong encryption of the communication, and mutual authentication of both communication sides, the client side (Cloud Connector) and the server side (SAP BTP).

The X.509 certificates which are used to authenticate the Cloud Connector and the SAP BTP subaccount are issued and controlled by SAP BTP. They are kept in secure storages in the Cloud Connector and in the cloud. Having encrypted and authenticated the tunnel, confidentiality and authenticity of the communication between the SAP BTP applications and the Cloud Connector is guaranteed.

Restricting Allowed Applications

As an additional level of control, the Cloud Connector optionally allows restricting the list of SAP BTP applications which are able to use the tunnel. This is useful in situations where multiple applications are deployed in a single SAP BTP subaccount while only particular applications require connectivity to on-premise systems.

Isolation on Subaccount Level

SAP BTP guarantees strict isolation on subaccount level provided by its infrastructure and platform layer. An application of one subaccount is not able to access and use resources of another subaccount.

Supported Protocols

The Cloud Connector supports inbound connectivity for HTTP and RFC, any other protocol is not supported.

- The payload sent via these protocols is encrypted on TLS/tunnel-level.
- For the route from the Cloud Connector to the on-premise systems, Cloud Connector administrators have the choice for each configured on-premise system whether to use HTTP, HTTPS, RFC or RFC over SNC.
- For HTTPS, you can configure a so-called system certificate in the Cloud Connector which is used for the trust relationship between the Cloud Connector and the connected on-premise systems.
- For RFC over SNC, you can configure an SNC PSE in the Cloud Connector respectively.

Principal Propagation

The Cloud Connector also supports principal propagation of the cloud user identity to connected on-premise systems (single sign-on). For this, the system certificate (in case of HTTPS) or the SNC PSE (in case of RFC) is mandatory to be configured and trust with the respective on-premise system must be established. Trust configuration, in particular for principal propagation, is the only reason to configure and touch an on-premise system when using it with the Cloud Connector.

Related Information

Configuring Principal Propagation [page 301]

1.4.5.4.2.3 Outbound Connectivity

The Cloud Connector supports the communication direction from the on-premise network to SAP BTP, using a database tunnel.

The database tunnel is used to connect local database tools via JDBC or ODBC to the SAP HANA DB or other databases onSAP BTP, for example, SAP Business Objects tools like Lumira, BOE or Data Services.

- The database tunnel only allows JDBC and ODBC connections from the Cloud Connector into the cloud. A reuse for other protocols is not possible.
- The tunnel uses the same security mechanisms as for the inbound connectivity:
  - TLS-encryption and mutual authentication

- Audit logging

To use the database tunnel, two different SAP BTP users are required:

- A platform user (member of the SAP BTP subaccount) establishes the database tunnel to the HANA DB.
- A HANA DB user is needed for the ODBC/JDBC connection to the database itself. For the HANA DB user, the role and privilege management of HANA can be used to control which actions he or she can perform on the database.

Related Information

Using Service Channels [page 443]

1.4.5.4.2.4 Audit Log

As audit logging is a critical element of an organization's risk management strategy, the Cloud Connector provides audit logging for the complete record of access between cloud and Cloud Connector as well as of configuration changes done in the Cloud Connector.

Integrity Check

The written audit log files are digitally signed by the Cloud Connector so that they can be checked for integrity (see also: Manage Audit Logs [page 508]).

Alerting

The audit log data of the Cloud Connector can be used to alert Cloud Connector administrators regarding unusual or suspicious network and system behavior.

Additional Use Cases

- The audit log data can provide auditors with information required to validate security policy enforcement and proper segregation of duties.
- IT staff can use the audit log data for root-cause analysis following a security incident.

Related Information

Audit Logging [page 508]

1.4.5.4.3 Cloud Infrastructure Layer

Infrastructure and network facilities of the SAP BTP ensure security on network layer by limiting access to authorized persons and specific business purposes.

Isolated Network

The SAP BTP landscape runs in an isolated network, which is protected from the outside by firewalls, DMZ, and communication proxies for all inbound and outbound communications to and from the network.

Sandboxed Environments

The SAP BTP infrastructure layer also ensures that platform services, like the SAP BTP Connectivity, and applications are running isolated, in sandboxed environments. An interaction between them is only possible over a secure remote communication channel.

1.4.5.4.4 Physical and Environmental Layer

Learn about data center security provided for SAP BTP Connectivity.

SAP BTP runs in SAP-hosted data centers which are compliant with regulatory requirements. The security measures include, for example:

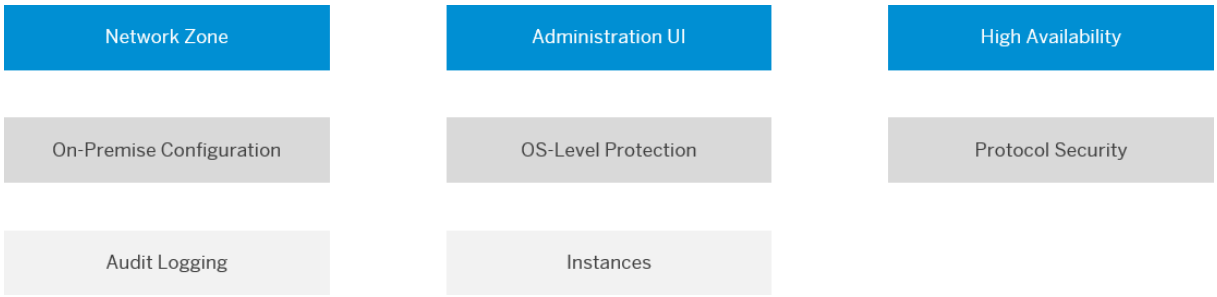
- strict physical access control mechanisms using biometrics, video surveillance, and sensors
- high availability and disaster recoverability with redundant power supply and own power generation

1.4.5.4.5 Security Guidelines

Find a checklist of recommended security measures for the Cloud Connector.

Topics

Hover over the elements for a description. Click an element to find the recommended actions in the table below.



- #unique_226/unique_226_Connect_42_network [page 527]
- #unique_226/unique_226_Connect_42_ui [page 528]
- #unique_226/unique_226_Connect_42_availability [page 529]
- #unique_226/unique_226_Connect_42_protocols [page 529]
- #unique_226/unique_226_Connect_42_os [page 528]
- #unique_226/unique_226_Connect_42_oP [page 529]
- #unique_226/unique_226_Connect_42_instances [page 530]
- #unique_226/unique_226_Connect_42_audit [page 529]

Recommended Actions

Topic	Description	Recommended Action
Network Zone Back to Topics [page 527]	Depending on the needs of the project, the Cloud Connector can be either set up in the **DMZ** and operated centrally by the IT department or set up in the **intranet** and operated by the line-of-business.	To access highly secure on-premise systems, operate the Cloud Connector centrally by the IT department and install it in the DMZ of the company network. Set up trust between the on-premise system and the Cloud Connector, and only accept requests from trusted Cloud Connectors in the system.

Topic	Description	Recommended Action
OS-Level Protection Back to Topics [page 527]	The Cloud Connector is a security-critical component that handles the inbound access from SAP BTP applications to systems of an on-premise network. **Methods to secure the operating system**, on which the Cloud Connector is running, should be applied.	Restrict access to the operating system on which the Cloud Connector is installed to the minimal set of users who should administrate the Cloud Connector.
		Use the machine which runs the Cloud Connector only for this purpose and don't reuse it for other scenarios.
		Use hard-drive encryption for the machine that runs the Cloud Connector. This ensures that the Cloud Connector configuration data cannot be read or modified by unauthorized users, even if they obtain access to the hard drive.
		Turn on the audit log on operating system level to monitor the file operations.
Administration UI Back to Topics [page 527]	After installation, the Cloud Connector provides an initial user name and password and forces the user (`Administrator`) to **change the password** upon initial logon.	Change the password of the *Administrator* user immediately after installation. Choose a strong password for the user (see also Recommendations for Secure Setup [page 266]).
		Configure a corporate LDAP system for the user management of the Cloud Connector administrator users. This guarantees that users of the Cloud Connector administration UI are named users and can be traced via the Cloud Connector audit log (see Use LDAP for Authentication [page 466]).
	You can access the Cloud Connector administration UI remotely via HTTPS. After installation, it uses a self-signed **X.509 certificate** as SSL server certificate, which is not trusted by default by Web browsers.	Exchange the self-signed X.509 certificate of the Cloud Connector administration UI by a certificate that is trusted by your company and the company's approved Web browser settings (see [Deprecated] Replace the Default SSL Certificate [page 272]).
		For high-security scenarios, limit the access to the Cloud Connector administration UI to localhost (see also Recommendations for Secure Setup [page 266]).

Topic	Description	Recommended Action
		Use a JVM that allows to limit the ciphers to a set considered safe (see Recommendations for Secure Setup [page 266]). A list of cipher suites, which are currently considered safe, is available in Use Secure Cipher Suites .
Audit Logging Back to Topics [page 527]	For end-to-end traceability of configuration changes in the Cloud Connector, as well as communication delivered by the Cloud Connector, **switch on audit logging** for productive scenarios.	Switch on audit logging in the Cloud Connector: set audit level to "All" (see Recommendations for Secure Setup [page 266] and Manage Audit Logs [page 508])
		Cloud Connector administrators must ensure that the audit log files are properly archived and are not lost, to conform to the local regulations.
		To gain end-to-end traceability, you should switch on audit logging also in the connected on-premise systems.
High Availability Back to Topics [page 527]	To guarantee high availability of the connectivity for cloud integration scenarios, run productive instances of the Cloud Connector in high availability mode, that is, with a **second (redundant) Cloud Connector** in place.	Use the high availability feature of the Cloud Connector for productive scenarios (see Install a Failover Instance for High Availability [page 470]).
Supported Protocols Back to Topics [page 527]	**HTTP**, **HTTPS**, **RFC** and **RFC over SNC** are currently supported as protocols for the communication direction from the cloud to on-premise. The route from the application VM in the cloud to the Cloud Connector is always encrypted. You can configure the route from the Cloud Connector to the on-premise system to be encrypted or unencrypted.	The route from the Cloud Connector to the on-premise system should be encrypted using TLS (for HTTPS) or SNC (for RFC). Trust between the Cloud Connector and the connected on-premise systems should be established (see Set Up Trust for Principal Propagation [page 301]).
Configuration of On-Premise Systems Back to Topics [page 527]	When configuring the access to an internal system in the Cloud Connector, map physical host names to virtual host names to **prevent exposure of information** on physical systems to the cloud.	Use hostname mapping of exposed on-premise systems in the access control of the Cloud Connector (see Configure Access Control (HTTP) [page 331] and Configure Access Control (RFC) [page 338]).

Topic	Description	Recommended Action
	When configuring the access to an internal system, **restrict access** to those resources which are actually required by the cloud applications. Do not expose the complete system.	Narrow access to on-premise systems to resources required by the relevant cloud applications in the access control of the Cloud Connector (see Configure Access Control (HTTP) [page 331] and Configure Access Control (RFC) [page 338]).
	To allow **access only for trusted applications** of your SAP BTP subaccount to on-premise systems, configure the list of trusted applications in the Cloud Connector.	Narrow the list of cloud applications which are allowed to use the on-premise tunnel to the ones that need on-premise connectivity (see Set Up Trust for Principal Propagation [page 301]).
Cloud Connector Instances Back to Topics [page 527]	You can connect a single Cloud Connector instance to multiple SAP BTP subaccounts. Subaccounts can be created by SAP BTP users as a self service. Different subaccounts are often used to **separate development, test and production**. Do not mix productive Cloud Connector usages with development or test scenarios.	Use different Cloud Connector instances to separate productive and non-productive scenarios.

Related Information

Recommendations for Secure Setup [page 266]
Secure the Activation of Traffic Traces [page 477]

1.4.5.5 Upgrade

Upgrade your Cloud Connector and avoid connectivity downtime during the update.

The steps for upgrading your Cloud Connector are specific to the operating system that you use. Previous settings and configurations are automatically preserved.

> ⚠ Caution
>
> Upgrade is supported only for *installer* versions, not for *portable* versions, see Installation [page 238]. Before upgrading, please check the Prerequisites [page 239] and make sure your environment fits the new version. We recommend that you create a Configuration Backup [page 460] before starting an upgrade.

Avoid Connectivity Downtime

If you have a single-machine Cloud Connector installation, a short downtime is unavoidable during the upgrade process. However, if you have set up a master and a shadow instance, you can perform the upgrade without downtime by executing the following procedure:

1. Shut down the shadow instance.
2. Perform the upgrade on the shadow instance. (Follow the relevant procedure below.)
3. Restart the shadow instance and wait until it has connected to the master instance.
4. Perform a *Switch Roles* operation by pressing the corresponding button in the master administration UI. The master instance has now changed into a shadow instance.

   > ⚠ Caution
   >
   > After upgrading the former shadow instance from a version prior to 2.13 and having switched its role to be the new master instance, reset high availability settings in *both* instances *now*, before continuing to upgrade the second instance from a version prior to 2.13 as well. The master-shadow connection must be re-established after both instances have been upgraded from versions prior to 2.13 to versions 2.13 or higher.

5. Shut down the new shadow instance and perform the upgrade procedure on it as well.
6. Restart the new shadow instance and wait until it has connected to the already upgraded current master instance.
7. Perform again the *Switch Roles* operation if you want the previous master instance to act as the new master instance again.

Result: Both instances have now been upgraded without connectivity downtime and without configuration loss.

For more information, see Install a Failover Instance for High Availability [page 470].

Microsoft Windows OS

1. Uninstall the Cloud Connector as described in Uninstallation [page 534] and make sure to retain the existing configuration.
2. Reinstall the Cloud Connector within the same directory. For more information, see Installation on Microsoft Windows OS [page 259].
3. Before accessing the administration UI, clear your browser cache to avoid any unpredictable behavior due to the upgraded UI.

Linux OS

1. Execute the following command: :

```
rpm -U com.sap.scc-ui-<version>.rpm
```

> **i Note**
>
> **Daemon extensions (as of Cloud Connector version 2.12.3)**
>
> All extensions to the daemon provided via `scc_daemon_extension.sh` mechanism will survive a version update. An upgrade to version 2.12.3 will already consider an existing file, even though previous versions were not supporting that feature.

2. Before accessing the administration UI, clear your browser cache to avoid any unpredictable behavior due to the upgraded UI.

1.4.5.6 Update the Java VM

How to update the Java VM used by the Cloud Connector.

Sometimes you must update the Java VM used by the Cloud Connector, for example, because of expired SSL certificates contained in the JVM, bug fixes, deprecated JVM versions, and so on.

- If you make a replacement in the same directory, shut down the Cloud Connector, upgrade the JVM, and restart the Cloud Connector when you are done.
- **If you change the installation directory** of the JVM, **follow the steps below** for your operating system.

Make sure that the JVM has been installed successfully.

> **i Note**
>
> A Java Runtime Environment (JRE) is not sufficient. You must use a JDK or SAP JVM.

Microsoft Windows OS

1. Make sure that the current user has administrative privileges.
2. Shutdown the Cloud Connector (for example, by stopping the corresponding Windows service or by double-clicking the *Stop SAP Cloud Connector* shortcut).
3. Open the *registry editor* (regedit32) and locate the following registry entry: `HKEY_LOCAL_MACHINE\SOFTWARE\SAP\Cloud Connector`.
4. Change the value `JavaHome` to reflect the installation directory of the new Java VM.

> **i Note**
>
> The *bin* subdirectory must not be part of the `JavaHome` value.

If the `JavaHome` value does not yet exist, create it here with a "String Value" (REG_SZ) and specify the full path of the Java installation directory, for example: `C:\Program Files\sapjvm`.

5. Close the registry editor and restart the Cloud Connector.

Linux OS

1. Open a shell as root user.
2. In this shell, set the environment variable `JAVA_HOME`, pointing to the installation directory of the new Java VM, for example, in tcsh:

```
setenv JAVA_HOME /opt/sap/sapjvm
```

3. Execute the command

```
System V init distributions: service scc_daemon stop
systemd distributions: systemctl stop scc_daemon
```

4. Execute the command

```
System V init distributions: /opt/sap/scc/daemon.sh reinstall
systemd distributions: /opt/sap/scc/daemon.sh reinstallSystemd
```

5. Execute the command

```
System V init distributions: service scc_daemon start
systemd distributions: systemctl start scc_daemon
```

Both Operating Systems

> **i Note**
>
> If you use your own CA certificates for the Email configuration (see Alerting [page 505]) or for LDAP (see Use LDAP for Authentication [page 466]), you must reimport them to the JVM trust store as described there.

After executing the above steps, the Cloud Connector should be running again and should have picked up the new Java version during startup. You can verify this by logging in to the Cloud Connector with your favorite browser, opening the *About* dialogue and checking that the field `<Java Details>` shows the version number and build date of the new Java VM. After you verified that the new JVM is indeed used by the Cloud Connector, delete or uninstall the old JVM.

1.4.5.7 Uninstallation

Uninstall an installer version or portable version of the Cloud Connector.

- If you have installed an installer variant of the Cloud Connector, follow the steps for your operating system to uninstall the Cloud Connector.
- To uninstall a developer version, proceed as described in section **Portable Variants**.

Microsoft Windows OS

1. In the Windows software administration tool, search for `Cloud Connector` (formerly named `SAP HANA cloud connector 2.x`).
2. Select the entry and follow the appropriate steps to uninstall it.
3. When you are uninstalling in the context of an upgrade, make sure to retain the configuration files.

Linux OS

To uninstall Cloud Connector 2.x, execute the following command:

```
rpm -e com.sap.scc-ui
```

> ⚠ Caution
>
> This command also removes the configuration files.

Mac OS X

There is no installer variant for Mac OS X, only a `portable` one.

Portable Variants

(Microsoft Windows OS, Linux OS, Mac OS X) If you have installed a portable version (zip or tgz archive) of the Cloud Connector, simply remove the directory in which you have extracted the Cloud Connector archive.

Related Information

Installation [page 238]

1.4.5.8 Frequently Asked Questions

Answers to the most common questions about the Cloud Connector.

Technical Issues

Does the Cloud Connector send data from on-premise systems to SAP BTP or the other way around?

The connection is opened from the on-premise system to the cloud, but is then used in the other direction.

An on-premise system is, in contrast to a cloud system, normally located behind a restrictive firewall and its services aren't accessible thru the Internet. This concept follows a widely used pattern often referred to as *reverse invoke proxy*.

Is the connection between the SAP BTP and the Cloud Connector encrypted?

Yes, by default, TLS encryption is used for the tunnel between SAP BTP and the Cloud Connector.

If used properly, TLS is a highly secure protocol. It is the industry standard for encrypted communication and also, for example, as a secure channel in HTTPS.

Keep your Cloud Connector installation updated and we will make sure that no weak or deprecated ciphers are used for TLS.

Can I use a TLS-terminating firewall between Cloud Connector and SAP BTP?

This is not possible. Basically, this is a desired man-in-the-middle attack, which does not allow the Cloud Connector to establish a mutual trust to the SAP BTP side.

Can I copy/clone a Cloud Connector installation and use it in parallel on a different machine?

This is not supported. You would face issues regularly, as those two instances will be considered as one, which is not expected by the cloud side.

If you just want to move the installation to a new machine, create a backup via the Configuration Backup [page 460] feature, create a new installation, import the backup, and discard the previous installation.

What is the oldest version of SAP Business Suite that's compatible with the Cloud Connector?

The Cloud Connector can connect an SAP Business Suite system version 4.6C and newer.

Which JRE versions are supported to run the Cloud Connector?

		Supported JRE Version		
		6	7	8
Cloud Connector Version	< 2.7.2	Yes	Yes	No
	= 2.7.2	Yes	Yes	Yes
	>= 2.8	No	Yes	Yes
	>=2.12.3	No	No	Yes

> **! Restriction**
>
> Support for Java 7 has been discontinued. For more information, see Prerequisites [page 241].

> **→ Tip**
>
> We recommend that you always use the latest supported JRE version.

> **⚠ Caution**
>
> Version 2.8 and later of theCloud Connector may have problems with ciphers in Google Chrome, if you use the JVM 7. For more information read this SCN Article.

Which configuration in the SAP BTP destinations do I need to handle the user management access to the Cloud User Store of the Cloud Connector?

See Configure an On-Premise User Store [page 441].

Is the Cloud Connector sufficient to connect the SAP BTP to an SAP ABAP back end or is SAP BTP Integration needed?

It depends on the scenario: For pure point-to-point connectivity to call on-premise functionality like BAPIs, RFCs, OData services, and so on, that are exposed via on-premise systems, the Cloud Connector might suffice.

However, if you require advanced functionality, for example, n-to-n connectivity as an integration hub, SAP BTP Integration – Process Integration is a more suitable solution. SAP BTP Integration can use the Cloud Connector as a communication channel.

How much bandwidth does the Cloud Connector consume?

The amount of bandwidth depends greatly on the application that is using the Cloud Connector tunnel. If the tunnel isn't currently used, but still connected, a few bytes per minute is used simply to keep the connection alive.

What happens to a response if there's a connection failure while a request is being processed?

The response is lost. The Cloud Connector only provides tunneling, it does not store and forward data when there are network issues.

Where should I install the Cloud Connector?

For productive instances, we recommend installing the Cloud Connector on a single purpose machine. This is relevant for Security [page 519]. For more details on which network zones to choose for the Cloud Connector setup, see Network Zones [page 250].

How many servers do I need to deploy the Cloud Connector?

We recommend that you use at least three servers, with the following purposes:

- Development
- Production master
- Production shadow

> **i Note**
>
> Do not run the production master and the production shadow as VMs inside the same physical machine. Doing so removes the redundancy, which is needed to guarantee high availability. A QA (Quality Assurance) instance is a useful extension. For disaster recovery, you will also need two additional instances; another master instance, and another shadow instance.

What are the hardware requirements to deploy the Cloud Connector?

See: Prerequisites [page 239].

Can I send push messages from an on-premise system to the SAP BTP through the Cloud Connector?

No, this is not supported by the Cloud Connector.

Is NTLM supported for authorization against the proxy server?

No, the Cloud Connector currently supports only basic authentication.

Which operating systems are supported by the Cloud Connector?

See Prerequisites [page 239].

Which processor architectures are supported by the Cloud Connector?

We currently support 64-bit operating systems running only on an x86-64 processor (also known as x64, x86_64 or AMD64), and for Linux also on the *PowerPC Little Endian* variant (also known as *ppc64le*).

See: Prerequisites [page 239].

Can I use the Cloud Connector without an ABAP back end?

Yes, you should be able to connect almost any system that supports the HTTP Protocol, to the SAP BTP, for example, Apache HTTP Server, Apache Tomcat, Microsoft IIS, or Nginx.

Can I authenticate with client certificates configured in SAP BTP destinations at HTTP services that are exposed via the Cloud Connector?

No, this is not possible. For client certificate authentication, an end-2-end TLS communication is required. This is not the case, because the Cloud Connector needs to inspect incoming requests in order to perform access control checks.

How can I do connection pooling for HTTP services that are exposed via the Cloud Connector?

The Cloud Connector itself does not perform connection pooling, but provides a 1-to-1 mapping for each logical connection received trough the tunnel.

By this mapping, a new connection to the backend system is opened, and kept open until closed either by the backend or by the client on cloud side.

The actual connection pooling is defined by the application client on cloud side:

- If a connection is re-used in the client library, it is re-used on the Cloud Connector side as well.
- If it is closed immediately, also the mapped one on Cloud Connector side will be closed immediately.

Administration

Are there Audit Logs for changes in the Cloud Connector?

Yes, find more details here: Manage Audit Logs [page 508].

Is it possible to split authorization?

No, currently there is only one role that allows complete administration of the Cloud Connector.

Can I configure multiple administrative subaccounts?

Yes, to enable this, you must configure an LDAP server. See: Use LDAP for Authentication [page 466].

How can I reset the Cloud Connector's administrator password when not using LDAP for authentication?

Visit https://tools.hana.ondemand.com/#cloud to download the portable version of the Cloud Connector. Extract the *users.xml* file in the *config* directory to the *config* directory of your Cloud Connector installation, then restart the Cloud Connector.

This resets the password and user name to their default values.

You can manually edit the file; however, we strongly recommend that you use the *users.xml* file.

How do I create a backup of the Cloud Connector configuration?

Starting with Cloud Connector version 2.11, you can use a dedicated backup feature, either from the administration UI (see Configuration Backup [page 460]) or via REST API (see Backup [page 398]).

Can I create a backup of the complete installation?

Yes, you can create an archive file of the installation directory to create a full backup. Before you restore from a backup, note the following:

- If you restore the backup on a different host, the UI certificate will be invalidated.
- Before you restore the backup, you should perform a “normal” installation and then replace the files. This registers the Cloud Connector at your operating systems package manager.

Why do I need a user ID during configuration?

This user opens the tunnel and generates the certificates that are used for mutual trust later on.

The user is not part of the certificate that identifies the Cloud Connector.

In both the Cloud Connector UI and in the SAP BTP cockpit, this user ID appears as the one who performed the initial configuration (even though the user may have left the company).

What happens to a Cloud Connector connection if the user who created the tunnel leaves the company?

This does not affect the tunnel, even if you restart the Cloud Connector.

What do changes in major or minor version numbers mean?

The semantics of Cloud Connector versions are explained in detail here.

For how long does SAP continue to support older Cloud Connector versions?

Each Cloud Connector version is supported for 12 months, which means the cloud side infrastructure is guaranteed to stay compatible with those versions.

After that time frame, compatibility is no longer guaranteed and interoperability could be dropped. Furthermore, after an additional 3 month, the next feature release published after that period will no longer support an upgrade from the deprecated version as a starting release.

What is the difference between “subaccount name” and “subaccount user”?

SAP BTP customers can purchase subaccounts and deploy applications into these subaccounts.

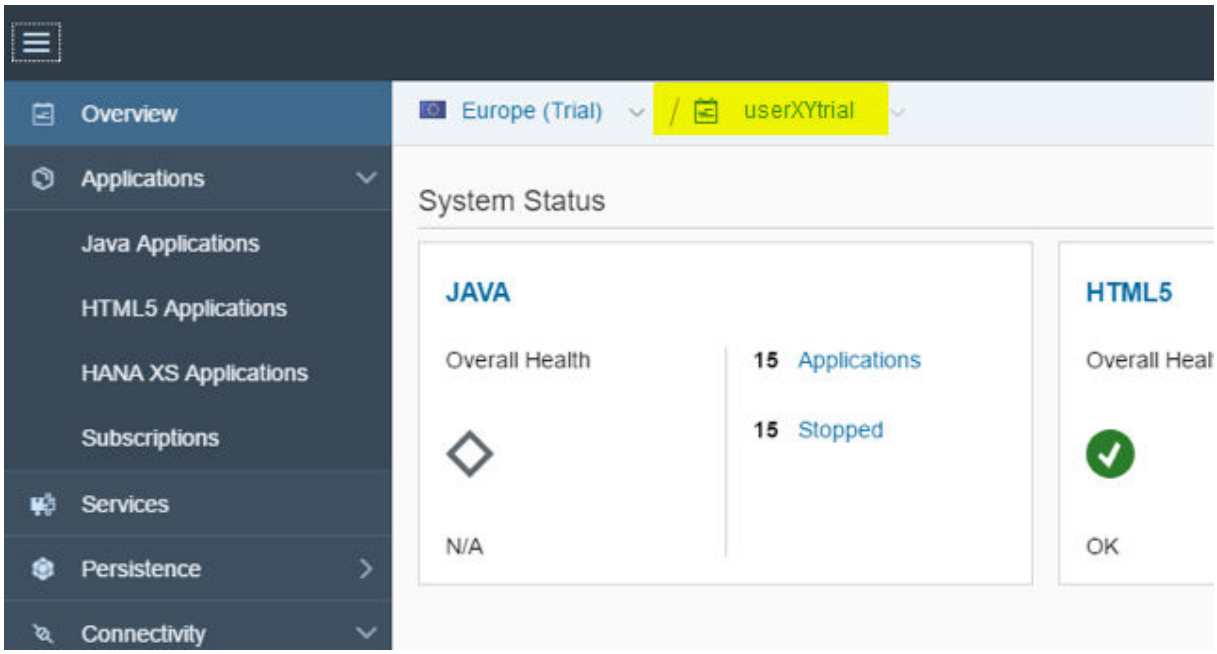
Additionally, there are users, who have a password and can log in to the cockpit and manage all subaccounts they have permission for.

- A single subaccount can be managed by multiple users, for example, your company may have several administrators.
- A single user can manage multiple subaccounts, for example, if you have multiple applications and want them (for isolation reasons) to be split over multiple subaccounts.

Find your account name by taking the following steps:

1. Open the SAP BTP cockpit.
2. Log in with your subaccount user.
3. You’ll see the subaccount name in the top left section of the screen.

For trial users, the account name is typically your user name, followed by the suffix “trial”:



Features

Does the Cloud Connector work with the SAP BTP Cloud Foundry environment?

As of version 2.10, the Cloud Connector can establish a connection to regions based on the SAP BTP Cloud Foundry environment. Newer regions, however, require a Cloud Connector version 2.11 or higher.

Does the Cloud Connector work with SAP S/4HANA Cloud?

As of version 2.10, the Cloud Connector offers a Service Channel to S/4HANA Cloud instances, given that they are associated with the respective SAP BTP subaccount. For more information, see Using Service Channels [page 443].

Also supported as of version 2.10: S/4HANA Cloud communication scenarios invoking HTTP services or remote-enabled function modules (RFMs) in on-premise ABAP systems.

Does the Cloud Connector work with the SAP BTP ABAP environment?

As of version 2.11, the Cloud Connector supports communication from and to the SAP BTP ABAP environment, when using the **Neo** Connectivity service. Using the **Cloud Foundry** Connectivity service requires a Cloud Connector version 2.12.3 or higher.

How do I bind multiple Cloud Connectors to one SAP BTP subaccount?

As of version 2.9, you can connect multiple Cloud Connectors to a single subaccount. This lets you assign multiple separate corporate network segments.

Those Cloud Connectors are distinguishable based on the location ID, which you must provide to the destination configuration on the cloud side.

> **i** Note
>
> During an upgrade, location IDs provided in earlier versions of the Cloud Connector are dropped to ensure that running scenarios are not disturbed.

Is WebSocket communication through the Cloud Connector supported?

Yes, this is possible as of version 2.12.

Is there any plan to add traffic management functionality in Cloud Connector?

No, this functionality is not currently planned.

Can I use the Cloud Connector for any protocol?

As of version 2.10, this is possible using the TCP channel of the Cloud Connector, if the client supports a SOCKS5 proxy to establish the connection. However, only the HTTP and RFC protocols currently provide an additional level of access control by checking invoked resources.

You can also use the Cloud Connector as a JDBC or ODBC proxy to access the HANA DB instance of your SAP BTP subaccount (service channel). This is sometimes called the "HANA Protocol".

Can I check the communication of the service channel?

No, the audit log monitors access only from SAP BTP to on-premise systems.

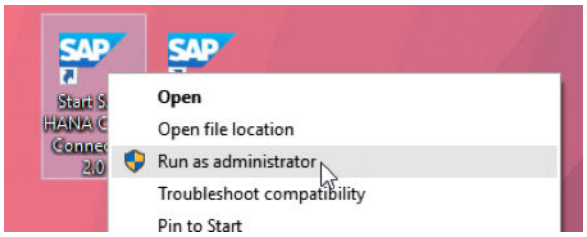
Troubleshooting

How do I fix the "Could not open Service Manager" error message?

You are probably seeing this error message due to missing administrator privileges. Right-click the cloud connector shortcut and select Run as administrator.

If you don't have administrator privileges on your machine you can use the portable variant of the Cloud Connector.

> **i** Note
>
> The portable variants of the Cloud Connector are meant for nonproductive scenarios only.



How do I set JAVA_HOME and PATH correctly?

For the portable versions, JAVA_HOME must point to the installation directory of your JRE, while PATH must contain the *bin* folder inside the installation directory of your JRE.

The installer versions automatically detect JVMs in these locations, as well as in other places.

When I try to open the Cloud Connector UI, Google Chrome opens a *Save as* dialog, Firefox displays some cryptic signs, and Internet Explorer shows a blank page, how do I fix this?

This happens when you try to access the Cloud Connector over HTTP instead of HTTPS. HTTP is the default protocol for most browsers.

Adding “https://” to the beginning of your URL should fix the problem. For localhost, you can use `https://localhost:8443/`.

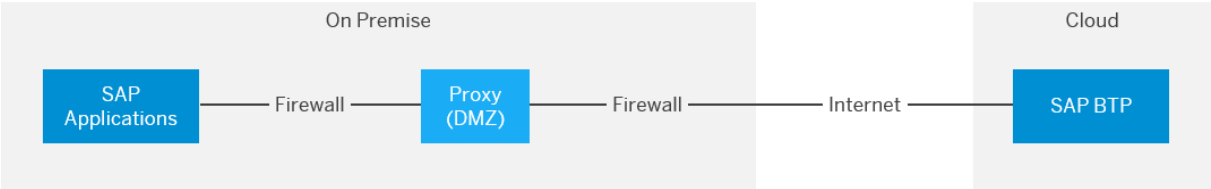
1.4.6 Connectivity via Reverse Proxy

An alternative approach compared to the SSL VPN solution that is provided by the Cloud Connector is to expose on-premise services and applications via a reverse proxy to the Internet. This method typically uses a reverse proxy setup in a customer's "demilitarized zone" (DMZ) subnetwork. The reverse proxy setup does the following:

- Acts as a mediator between SAP BTP and the on-premise services
- Provides the services of an Application Delivery Controller (ADC) to, for example, encrypt, filter, route, or check inbound traffic

The figure below shows the minimal overall network topology of this approach.

On-premise services that are accessible via a reverse proxy are callable from SAP BTP like other HTTP services available on the Internet. When you use destinations to call those services, make sure the configuration of the `ProxyType` parameter is set to **`Internet`**.



Advantages

Depending on your scenario, you may benefit from the reverse proxy:

- Network infrastructure (such as a reverse proxy and ADC services): since it already exists in your network landscape, you can reuse it to connect to SAP BTP. There's no need to set up and operate new components on your (customer) side.
- A reverse proxy is independent of the cloud solution you are using.
- It acts as single entry point to your corporate network.

Disadvantages

- The reverse proxy approach leaves exposed services generally accessible via the Internet. This makes them vulnerable to attacks from anywhere in the world. In particular, **Denial-of-Service attacks** are possible and difficult to protect against. To prevent attacks of this type and others, you must implement the highest security in the DMZ and reverse proxy. For the productive deployment of a hybrid cloud/on-

premise application, this approach usually requires intense involvement of the customer's IT department and a longer period of implementation.

- If the reverse proxy allows filtering, or restricts accepted source IP addresses, you can set only one IP address to be used for all SAP BTP outbound communications.
  A reverse proxy does not exclusively restrict the access to cloud applications belonging to a customer, although it does filter any callers that are not running on the cloud. Basically, any application running on the cloud would pass this filter.
- The SAP-proprietary RFC protocol is supported only if WebSocket RFC can be used for communication with the ABAP system. WebSocket RFC is available as of S/4HANA release 1909. A cloud application cannot call older on-premise ABAP systems directly without using application proxies on top of ABAP in between.
- No easy support of principal propagation authentication, which lets you forward the cloud user identity to on-premise systems.
- You cannot implement projects close to your line of business (LoB).

> **i Note**
>
> Using the Cloud Connector mitigates all of these issues. As it establishes the SSL VPN tunnel to SAP BTP using a reverse invoke approach, there is no need to configure the DMZ or external firewall of a customer network for inbound traffic. Attacks from the Internet are not possible. With its simple setup and fine-grained access control of exposed systems and resources, the Cloud Connector allows a high level of security and fast productive implementation of hybrid applications. It also supports multiple application protocols, such as HTTP and RFC.

1.4.7 Connectivity Support

Support information for SAP BTP Connectivity and the Cloud Connector.

Troubleshooting

Locate the problem or error you have encountered and follow the recommended steps:

- Frequently Asked Questions [page 535] (Cloud Connector)
- Administration [page 464] (Cloud Connector)
- Cloud Connectivity: Guided Answers
- Getting Support (SAP Support Portal, SAP BTP community)

SAP Support Information

If you cannot find a solution to your issue, collect and provide the following specific, issue-relevant information to SAP Support:

- The Java EE code that throws an error (if any)
- A screenshot of the error message for the failed operation or the error message from the `HttpResponse` body
- Access credentials for your cloud or on-premise location

You can submit this information by creating a customer ticket in the SAP CSS system using the following components:

Component	Purpose
Connectivity Service	
BC-CP-CON	For *cloud-side* issues with **cloud to on-premise** connectivity, where: • The **environment is unknown** or • The issue is **not related** to a **specific environment**
BC-CP-CON-CF	For *cloud-side* issues with **cloud to on-premise** connectivity in the SAP BTP **Cloud Foundry environment**.
BC-CP-CON-S4HC	For *cloud-side* issues with **cloud to on-premise** connectivity in an **S/4HANA Cloud** system.
BC-CP-CON-K8S-PROXY	For cloud-side issues with **cloud to on-premise** connectivity in a Kubernetes cluster (or Kubernetes-based product), using the connectivity proxy software component.
BC-CP-CON-ABAP	For *cloud-side* issues with **cloud to on-premise** connectivity in the SAP BTP **ABAP environment**.
BC-NEO-CON	For *cloud-side* issues with **cloud to on-premise** connectivity in the SAP BTP **Neo environment**.
Destinations	
BC-CP-DEST	For issues with **destination configurations**, where: • The **environment is unknown** or • The issue is **not related** to a **specific environment**
BC-CP-DEST-CF	For **general issues** with the Destination service in the SAP BTP **Cloud Foundry environment**, like: • REST API • Instance creation, etc.
BC-CP-DEST-CF-CLIBS	For **client library issues** with the Destination service in the SAP BTP **Cloud Foundry environment**.
BC-CP-DEST-CF-TOOLS	For issues with the management of **destination configurations via tools** like the SAP BTP cockpit (**Cloud Foundry environment**).

Component	Purpose
BC-CP-DEST-NEO	For issues with **destination configurations** or: • Management tools • Client libraries, etc. related to destinations in the SAP BTP **Neo environment**.
Cloud Connector	
BC-MID-SCC	For connectivity issues related to installing and configuring the **Cloud Connector**, configuring tunnels, connections, and so on.

If you experience a more serious issue that cannot be resolved using only traces and logs, SAP Support may request access to the Cloud Connector. Follow the instructions in these SAP notes:

- To provide access to the Administration UI via a browser, see 592085.
- To provide SSH access to the operating system of the Linux machine on which the connector is installed, see 1275351.

Related Information

Release and Maintenance Strategy [page 546]

1.4.7.1 Release and Maintenance Strategy

Find information about SAP BTP Connectivity releases, versioning and upgrades.

Release Cycles

Updates of the Connectivity service are published as required, within the regular, bi-weekly SAP BTP release cycle.

New releases of the Cloud Connector are published when new features or important bug fixes are delivered, available on the Cloud Tools page.

Cloud Connector Versions

Cloud Connector versions follow the `<major>.<minor>.<micro>` versioning schema. The Cloud Connector stays fully compatible within a major version. Within a minor version, the Cloud Connector will stay with the same feature set. Higher minor versions usually support additional features compared to lower minor versions. Micro versions generally consist of patches to a `<master>.<minor>` version to deliver bug fixes.

For each supported major version of the Cloud Connector, only one `<major>.<minor>.<micro>` version will be provided and supported on the Cloud Tools page. This means that users must upgrade their existing Cloud Connectors to get a patch for a bug or to make use of new features.

Cloud Connector Upgrade

New versions of the Cloud Connector are announced in the Release Notes of SAP BTP. We recommend that Cloud Connector administrators regularly check the release notes for Cloud Connector updates. New versions of the Cloud Connector can be applied by using the Cloud Connector upgrade capabilities. For more information, see Upgrade [page 530].

> **i Note**
>
> We recommend that you first apply upgrades in a test landscape to validate that the running applications are working.

There are no manual user actions required in the Cloud Connector when the SAP BTP is updated.

1.5 What is Document Service

Store and manage all your documents.

SAP Document Service helps you manage your business documents. It is based on the OASIS industry-standard CMIS and offers versioning, hierarchies, access control, and document management capabilities.

Environment

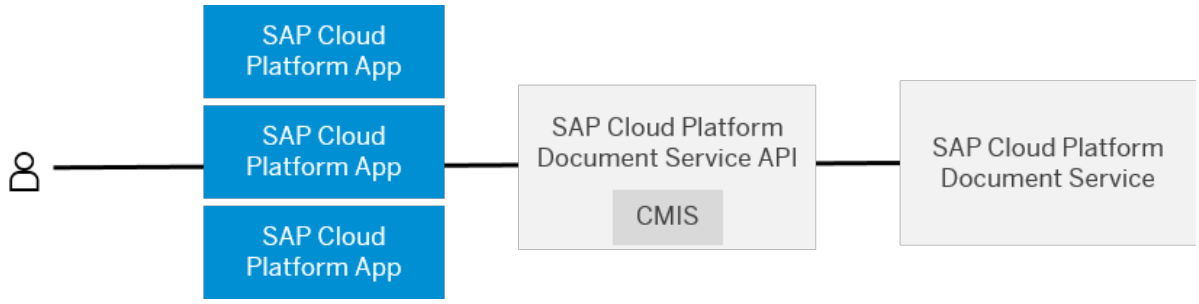
This service runs in the Neo environment.

Features

Store and retrieve unstructured content	Achieve more with the persistent content storage that provides a standardized interface for content on the OASIS industry standard CMIS.

Use Client API for Java applications	Use the client API on top of the protocol for easier consumption of your stored data. This is a OpenCMIS API provided by Apache Chemistry.
Achieve more	Structure your content in a meaningful way using folder hierarchies. Version your content, manage check-in and checkout of documents for collaboration and track the history. Further, retrieve the metadata attached to content using a query language like SQL.

General



The document service is an implementation of the CMIS standard and is the primary interface to a reliable and safe store for content on SAP BTP.

The CMIS standard defines:

- A domain model and service bindings that can be used by applications to work with a content management repository
- An abstraction layer for controlling diverse document management systems and repositories using Web protocols

CMIS provides a common data model covering typed files and folders with generic properties that can be set or read. There is a set of services for adding and retrieving documents (called objects). CMIS defines an access control system, a checkout and version control facility, and the ability to define generic relations. CMIS defines the following protocol bindings, which use WSDL with Simple Object Access Protocol (SOAP) or Representational State Transfer (REST):

- The Atom Publishing (AtomPub) Protocol
- The JavaScript Object Notation (JSON) format

The consumption of CMIS-enabled document repositories is easy using the Apache Chemistry libraries. Apache Chemistry provides libraries for several platforms to consume CMIS using Java, PHP, .Net, or Python. The subproject OpenCMIS, which includes the CMIS Java implementation, also includes tools around CMIS, like the CMIS Workbench, which is a desktop client for CMIS repositories for developers.

Since the SAP Document service API includes the OpenCMIS Java library, applications can be built on SAP BTP that are independent of a specific content repository.

Restrictions

The SOAP (Web services) binding is not supported.

The following features, which are defined in the OASIS CMIS standard, are supported with restrictions:

- Versioning: Only major versions are supported
- Versioning: No support for check-in comments
- Query: Only metadata searches, no joins and no type aliases

The following CMIS features are not yet supported:

- Multifiling
- Policies
- Relationships
- Change logs

There is a limit for the properties of a document:

- For searchable properties, a maximum of 100 values with a maximum of 5,000 characters is allowed.
- For non-searchable properties, a maximum of 1,000 values with a maximum of 50,000 characters is allowed.
- Maximal allowed length of one property is 4,500 characters.

The document service can be utilized up to the following maximum limits:

- Maximum total of 5 million objects (documents or folders) per tenant repository
- Maximum total of 100 tenant repositories per subaccount
- Maximum duration of 60 seconds of a single CMISQL query
- All requests are subject to concurrent rate control and spike limiting

If you expect to reach one or the other limitation, we recommend that you open a support ticket on `BC-NEO-ECM-DS` and describe your scenario.

Overview

Applications access the document service using the OASIS standard protocol Content Management Interoperability Services (CMIS). Java applications running on SAP BTP can easily consume the document service using the provided client library. Since the document service is exposed using a standard protocol, it can also be consumed by any other technology that supports the CMIS protocol.

Related Information

Consuming the Document Service

Consuming the Document Service (Java) [page 550]
Consume Document Service (HTML5 Applications) [page 598]

Managing the Document Service

Managing Repositories in the Cockpit [page 599]

Managing a Repository with Console Client Commands [page 602]

General Information on CMIS

OASIS Page on CMIS

Apache Chemistry Page

OASIS Page with link to CMIS-v1.1pdf

1.5.1 Consuming the Document Service (Java)

Use the SAP Document service to store unstructured or semi-structured data in the context of your SAP BTP application.

Introduction

Many applications need to store and retrieve unstructured content. Traditionally, a file system is used for this purpose. In a cloud environment, however, the usage of file systems is restricted. File systems are tied to individual virtual machines, but a Web application often runs distributed across several instances in a cluster. File systems also have limited capacity.

The document service offers persistent storage for content and provides additional functionality. It also provides a standardized interface for content using the OASIS CMIS standard.

Related Information

Basic Concepts (Java) [page 550]

Handling CMIS Metadata [page 570]

Create Sample Applications (Java) [page 557]

1.5.1.1 Basic Concepts (Java)

The following sections describe the basic concepts of the SAP Document service.

- Client API (Java) [page 551]
- Documents and Folders (Java) [page 553]
- Deployment Options [page 555]
- Data Isolation (Java) [page 555]
- Consistency Model (Java) [page 557]

In the coding and the coding samples, `ecm` is used to refer to the document service. Therefore, for example, the document service API is called `ecm.api`.

1.5.1.1.1 Client API (Java)

The SAP Document service is exposed using the OASIS standard protocol Content Management Interoperability Service (CMIS).

The CMIS standard defines the protocol level (SOAP, AtomPub, and JSON based protocols). The SAP BTP provides a document service client API on top of this protocol for easier consumption. This API is the Open Source library OpenCMIS provided by the Apache Chemistry Project.

Related Information

Apache Chemistry Project
ecm.api
What is Document Service [page 547]

1.5.1.1.2 Repositories (Java)

To manage documents in the SAP Document service, you need to connect an application to a repository of the document service.

A repository is the document store for your application. It has a unique name with which it can later be accessed, and it is secured using a key provided by the application. Only applications that provide this key are allowed to connect to this repository.

You can manage repositories in several ways:

- In the cockpit, see Managing Repositories in the Cockpit [page 599].
- Programmatically using the `createRepository(repositoryOptions)` method of the `EcmService`, see Managing Repositories Programmatically (Java) [page 552].
- Using console client commands, see Managing a Repository with Console Client Commands [page 602].

> **i Note**
>
> As a repository has a certain storage footprint in the back end, the total amount of repositories for each subaccount is limited to 100. When you create repositories, for example, for testing, make sure that these repositories are deleted after a test is finished to avoid reaching the limit. Should your use case require more than 100 repositories per subaccount, please create a support ticket.

> **i Note**
>
> Due to the tenant isolation in SAP BTP, the document service cockpit cannot access or view repostories you create in SAP Document Center or vice versa.

1.5.1.1.2.1 Managing Repositories Programmatically (Java)

You can manage a repository using the application's program. In this way, you can create, edit, delete, and connect the repository.

Related Information

Create Repositories Programmatically (Java) [page 552]
Connect to Repositories (Java) [page 553]

1.5.1.1.2.1.1 Create Repositories Programmatically (Java)

You can create a repository with the `createRepository(repositoryOptions)` method of the `EcmService` (document service).

Procedure

Use the `createRepository(repositoryOptions)` method and define the properties of the repository.

The following code snippet shows how to create a repository where uploaded files are scanned for viruses:

```
RepositoryOptions options = new RepositoryOptions();
options.setUniqueName("myrepository");
options.setRepositoryKey("1234567890");
options.setVirusScannerEnabled(true);
EcmService.createRepository(options);
```

Related Information

Alternative Ways to Create Repositories
create-ecm-repository
Create Repositories (Cockpit) [page 599]
Connecting Your Repository to an Application
Create Sample Applications (Java) [page 557]

1.5.1.1.2.1.2 Connect to Repositories (Java)

Your application must be connected to a repository.

Context

There are many ways to connect to a repository. For more information, see the API Documentation [page 1147] and *Reuse OpenCmis Session Objects* in Performance Tips (Java) [page 594].

Procedure

To connect to an existing repository, use the `connect(uniqueName, key)` method.

Once you are connected to the repository, you get an OpenCMIS session object to manage documents and folders in the connected repository.

1.5.1.1.3 Documents and Folders (Java)

Probably the most common use case is to create documents and folders in a repository. Every repository in CMIS has a root folder. Once you have received a `Session`, you can create the root folder using the following syntax:

```
// get the root folder of the repository
Folder root = openCmisSession.getRootFolder();
```

Once you have a root folder, you can create other folders or documents. In the CMIS domain model, all CMIS objects are typed. Therefore, you have to provide type information for each object you create. The types carry the metadata for an object. The metadata is passed in a property `map`. Some properties are mandatory, others are optional. You have to provide at least an object type and a name. For properties defined in the standard, OpenCMIS has predefined constants in the `PropertyIds` class.

To create a document in the root folder, enter the following syntax:

```
Map<String, String> newFolderProps = new HashMap<String, String>();
newFolderProps.put(PropertyIds.OBJECT_TYPE_ID, "cmis:folder");
newFolderProps.put(PropertyIds.NAME, "MyFirstFolder");
root.createFolder(newFolderProps);
```

To create a document with content, provide a map of properties. In addition, create a `ContentStream` object carrying a Java `InputStream` plus some additional information for the content, like `Content-Type` and file name.

```
Map<String, String> properties = new HashMap<String, String>();
properties.put(PropertyIds.OBJECT_TYPE_ID, "cmis:document");
properties.put(PropertyIds.NAME, "HelloWorld.txt");
```

```
byte[] helloContent = "Hello World!".getBytes( "UTF-8");
InputStream stream = new ByteArrayInputStream(helloContent);
ObjectFactory factory = openCmisSession.getObjectFactory();
ContentStream contentStream = factory.createContentStream("HelloWorld.txt",
    helloContent. length, "text/plain; charset=UTF-8", stream);
```

To create the document, enter the following syntax:

```
Document myDocument = root.createDocument(properties, contentStream,
VersioningState.NONE);
```

Get the ID for later retrieval of the document:

```
String id = myDocument.getId();
```

Getting Children

To get the children of a folder, you can use the following code:

```
Folder root = openCmisSession.getRootFolder();
ItemIterable<CmisObject> children = root.getChildren();
for (CmisObject o : children) {
    System.out.print("Name: " + o.getName());
    if (o instanceof Folder) {
        System.out.println(", type: Folder, createdBy: " + o.getCreatedBy());
    }
    else {
        Document doc = (Document) o;
        System.out.println.println(", type: Document, createdBy: " +
o.getCreatedBy() +
            " filesize: "+ doc.getContentStreamLength() + " bytes");
    }
}
```

Retrieving a Document

To retrieve a document, you can use the following code:

```
Document document = (Document) openCmisSession.getObject(id);
Property<String> p = document.getProperty(PropertyIds.NAME);
System.out.println("Name: " + p.getValue());
// or use System.out.println("Name: " + document.getName());
```

To get the content, use the following code:

```
InputStream stream = document.getContentStream().getStream();
```

You can also retrieve a document using its path with the `getObjectByPath()` method.

> → Tip
>
> We recommend that you retrieve objects by ID and not by path. IDs are kept stable even if the object is moved. Retrieving objects by IDs is also faster than retrieving objects by paths.

1.5.1.1.4 Deployment Options

Before your application can use the document service, the application must be able to access and consume the service.

There are several ways in which your application can access the document service:

- Any application deployed on SAP BTP as a Java Web application can consume the document service.
- During the development phase, you can also use the document service in the SAP BTP local runtime.
  As a prerequisite for local development, you need an installation of the MongoDB on your machine. See Create Sample Applications (Java) [page 557].
- You can also use the document service from an application running outside SAP BTP.
  This requires a special application running on SAP BTP acting as a bridge between the external application and the document service. This application is called a "proxy bridge". For more information, see Build a Proxy Bridge [page 562].

Related Information

http://chemistry.apache.org/

1.5.1.1.5 Data Isolation (Java)

User Management

The service treats user names as opaque strings that are defined by the application. All actions in the document service are executed in the context of this named user or the currently logged-on user. That is, the service sets the `cmis:createdBy` and `cmis:lastModifiedBy` properties to the provided user name. The service also uses this user name to evaluate access control lists (ACLs). For more information, see the CMIS specification. The document service is not connected to a user management system and, therefore, does not perform any user authentication.

Repository Naming and Data Isolation

Repositories are identified either by their unique name or by their ID. The unique name is a human-readable name that should be constructed with Java package-name semantics, for example, `com.foo.MySpecialRepository`, to avoid naming conflicts. Repositories in the document service are secured by a key provided by the application. When a repository is created, a key must be supplied. Any further attempts to connect to this repository only succeed if the key provided by the connecting application matches the key that was used to create the repository. Therefore, this key must be stored in a secure manner, for

example, using the Java KeyStore. It is, however, up to the application to decide whether to share this key with other applications from the same subaccount to implement data-sharing scenarios.

Multiple applications can access the same repository. However, applications can only connect to the same repository using the unique name assigned to this repository if they are deployed within the same subaccount as the application that created the repository. In contrast, applications that are deployed in a different subaccount cannot access this repository. A consequence of having repositories isolated within a subaccount is that data cannot be shared across different subaccounts.

Example of Subaccount Isolation

Repository ABC is created when $Application_1$ is deployed in $Subaccount_1$. $Application_2$ is located in the same $Subaccount_1$ as $Application_1$; therefore, $Application_2$ can also access the same repository using its unique name ABC. $Application_3$ is deployed in $Subaccount_2$. $Application_3$ calls a repository that has the same unique name ABC as the other repository that belongs to $Subaccount_1$. However, $Application_3$ cannot access the ABC repository that belongs to $Subaccount_1$ using the identical unique name, because the repositories are isolated within the subaccount. Therefore, $Application_3$ in $Subaccount_2$ connects to another ABC repository that belongs to $Subaccount_2$. In summary, a repository can only be accessed by applications that are deployed in the same subaccount as the application that created the repository.

Multitenancy

The document service supports multitenancy and isolates data between tenants. Each application consuming the document service creates a repository and provides a unique name and a secret key. The document service creates the repository internally in the context of the tenant using the application. While the repository name uniquely identifies the repository, an internal ID is created for the application for each tenant. This ID identifies the storage area containing all the data for the tenant in this repository. An application that uses the document service in this way has multitenancy support. No additional logic is required at the application level.

> → Tip
>
> One document service session is always bound to one tenant and to one user. If you create the session only once, then store it statically, and finally reuse it for all subsequent requests, you end up in the tenant where you first created the document service session. That is: You do not use multitenancy.
>
> We recommend that you create one document service session per tenant and cache these sessions for future reuse. Make sure that you do not mix up the tenants on your side.
>
> If you expect a high load for a specific tenant, we recommend that you create a pool of sessions for that tenant. A session is always bound to a particular server of the document service and this will not scale. If you use a session pool, the different sessions are bound to different document service servers and you will get a much better performance and scaling.

Related Information

Content Management Interoperability Services (CMIS) Version 1.1
Developing Multitenant Applications in the Neo Environment [page 898]

1.5.1.1.6 Consistency Model (Java)

Changes to the data are visible to other ECM sessions only with some delay.

If the data of a repository is changed, for example, by creating, modifying, or deleting documents or folders, an `ECM` session fetched from the class `EcmFactory` is used. Only subsequent read operations of the same session (session-based read your own writes) see such changes immediately. All other sessions see such changes only after some time (eventual consistency), usually within a few seconds but in case of heavy load scenarios also after a longer delay.

1.5.1.2 Create Sample Applications (Java)

Prerequisites

- You have downloaded and configured the SAP Eclipse platform. For more information, see Setting Up the Development Environment [page 845].
- You have created a HelloWorld Web application as described in .
- You have downloaded the SDK used for local development.
- You have installed MongoDB as described in Setup Local Development [page 561].

Context

This tutorial describes how you extend the HelloWorld Web application so that it uses the SAP Document service for managing unstructured content in your application. You test and run the Web application on your local server and the SAP BTP.

> **i Note**
>
> For historic reasons, `ecm` is used to refer to the document service in the coding and the coding samples.

Procedure

1. Connect the HelloWorld Web application to the document service.

The document service client library is used to connect to the document service. The library connects to the local or central document service and returns an authenticated OpenCMIS session. If you are running your application locally in the Eclipse IDE, the document service client library connects to a local document service of the SAP BTP SDK that is connected to your local MongoDB. If your application is deployed on SAP BTP, the document service client library connects to the document service running in this region.

2. If your application needs authenticated users and these users should be automatically propagated to the document service, configure your Web application to enable user authentication.
   a. Expand the *HelloWorld/WebContent/WEB-INF* node.
   b. Select the `web.xml` file and choose *Open* from the context menu.
   c. Enable authentication for your application.

   For more information about authentication, see Authentication [page 1678].
3. Connect to the document service and create a folder and a document.
   a. Expand the *HelloWorld/Java Resources/src/hello* node.
   b. Select the `HelloWorldServlet.java` file and, choose *Open* from the context menu.
   c. Add the following code to the `HelloWorldServlet.java`.

```
package hello;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.chemistry.opencmis.client.api.CmisObject;
import org.apache.chemistry.opencmis.client.api.Document;
import org.apache.chemistry.opencmis.client.api.Folder;
import org.apache.chemistry.opencmis.client.api.ItemIterable;
import org.apache.chemistry.opencmis.client.api.Session;
import org.apache.chemistry.opencmis.commons.PropertyIds;
import org.apache.chemistry.opencmis.commons.data.ContentStream;
import org.apache.chemistry.opencmis.commons.enums.VersioningState;
import
org.apache.chemistry.opencmis.commons.exceptions.CmisNameConstraintViolatio
nException;
import
org.apache.chemistry.opencmis.commons.exceptions.CmisObjectNotFoundExceptio
n;
import com.sap.ecm.api.RepositoryOptions;
import com.sap.ecm.api.RepositoryOptions.Visibility;
import com.sap.ecm.api.EcmService;
import javax.naming.InitialContext;
/**
 * Servlet implementation class HelloWorldServlet
 */
public class HelloWorldServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;
  /**
   * @see HttpServlet#HttpServlet()
   */
  public HelloWorldServlet() {
    super();
  }
  /**
   * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
   *      response)
   */
  protected void doGet(HttpServletRequest request,
```

```
                        HttpServletResponse response) throws
ServletException, IOException {
    response.getWriter().println("<html><body>");
    try {
      // Use a unique name with package semantics e.g. com.foo.MyRepository
      String uniqueName = "com.foo.MyRepository";
      // Use a secret key only known to your application (min. 10 chars)
      String secretKey = "my_super_secret_key_123";
      Session openCmisSession = null;
        InitialContext ctx = new InitialContext();
        String lookupName = "java:comp/env/" + "EcmService";
        EcmService ecmSvc = (EcmService) ctx.lookup(lookupName);
      try {
        // connect to my repository
        openCmisSession = ecmSvc.connect(uniqueName, secretKey);
      }
      catch (CmisObjectNotFoundException e) {
        // repository does not exist, so try to create it
        RepositoryOptions options = new RepositoryOptions();
        options.setUniqueName(uniqueName);
        options.setRepositoryKey(secretKey);
        options.setVisibility(Visibility.PROTECTED);
        ecmSvc.createRepository(options);
        // should be created now, so connect to it
        openCmisSession = ecmSvc.connect(uniqueName, secretKey);
      }
      response.getWriter().println(
        "<h3>You are now connected to the Repository with Id "
        + openCmisSession.getRepositoryInfo().getId()
        + "</h3>");
      // access the root folder of the repository
      Folder root = openCmisSession.getRootFolder();
      // create a new folder
      Map<String, String> newFolderProps = new HashMap<String, String>();
      newFolderProps.put(PropertyIds.OBJECT_TYPE_ID, "cmis:folder");
      newFolderProps.put(PropertyIds.NAME, "MyFolder");
      try {
        root.createFolder(newFolderProps);
      } catch (CmisNameConstraintViolationException e) {
        // Folder exists already, nothing to do
      }
      // create a new file in the root folder
      Map<String, Object> properties = new HashMap<String, Object>();
      properties.put(PropertyIds.OBJECT_TYPE_ID, "cmis:document");
      properties.put(PropertyIds.NAME, "HelloWorld.txt");
      byte[] helloContent = "Hello World!".getBytes("UTF-8");
      InputStream stream = new ByteArrayInputStream(helloContent);
      ContentStream contentStream = openCmisSession.getObjectFactory()
                                    .createContentStream("HelloWorld.txt",
                                    helloContent.length, "text/plain;
charset=UTF-8", stream);
      try {
        root.createDocument(properties, contentStream,
VersioningState.NONE);
      } catch (CmisNameConstraintViolationException e) {
        // Document exists already, nothing to do
      }
      // Display the root folder's children objects
      ItemIterable<CmisObject> children = root.getChildren();
      response.getWriter().println("The root folder of the repository with
id " + root.getId()
                                + " contains the following objects:<ul>");
      for (CmisObject o : children) {
        response.getWriter().print("<li>" + o.getName());
        if (o instanceof Folder) {
          response.getWriter().println(" createdBy: " + o.getCreatedBy() +
"</li>");
        } else {
```

```
          Document doc = (Document) o;
          response.getWriter().println(" createdBy: " + o.getCreatedBy() +
" filesize: "
                                       + doc.getContentStreamLength() + "
bytes"
                                       + "</li>");
        }
      }
      response.getWriter().println("</ul>");
    } catch (Exception e) {
      throw new ServletException(e);
    } finally {
      response.getWriter().println("</body></html>");
    }
  }
  /**
   * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse
   *      response)
   */
  protected void doPost(HttpServletRequest request,
                        HttpServletResponse response) throws
ServletException, IOException {
    // TODO Auto-generated method stub
  }
}
```

For more information about using the OpenCMIS API, see the Apache Chemistry documentation.

During execution, this servlet executes the following steps:

1. It connects to a repository. If the repository does not yet exist, the servlet creates the repository.
2. It creates a subfolder.
3. It creates a document.
4. It displays the children of the root folder.

4. Add the resource reference description to the `web.xml` file.

> **i Note**
>
> The document service is consumed by defining a resource in your `web.xml` file and by using JNDI lookup to retrieve an instance of the `com.sap.ecm.api.EcmService` class. Once you have established a connection to the document service, you can use one of the `connect(…)` methods to get a CMIS session (`org.apache.chemistry.opencmis.client.api.Session`). A few examples of how to use the OpenCMIS Client API from the Apache Chemistry project are described below. For more information, see the Apache Chemistry page.

   a. In the *Project Explorer* view, expand the *HelloWorld/WebContent/WEB-INF* node.
   b. Select the `web.xml` file and choose *Open With* > *Text Editor* from the context menu.
   c. Insert the following content after the `<servlet-mapping>` elements.

```
<resource-ref>
   <res-ref-name>EcmService</res-ref-name>
   <res-type>com.sap.ecm.api.EcmService</res-type>
</resource-ref>
```

5. Test the Web application locally or in the SAP BTP. For testing, proceed as described in or linked below.

Related Information

Authentication [page 1678]
http://chemistry.apache.org/java/opencmis.html
http://chemistry.apache.org/
http://chemistry.apache.org/java/developing/guide.html
http://chemistry.apache.org/java/0.13.0/maven/apidocs/
http://chemistry.apache.org/java/examples/index.html

1.5.1.3 Setup Local Development

To use the document service in a Web application, download the SDK and install the MongoDB database.

Context

> ⚠ Caution
>
> The local document service emulation is deprecated as of 5 March 2018. Support will be discontinued after 5 July 2018. This does not affect the availability of the document service running on SAP BTP, but only its local emulation that is part of the SDK.
>
> We recommend to either deploy applications consuming the document service to SAP BTP, or to consume a cloud-located repository locally as described in Access from External Applications [page 562]. This explains how to access a document service repository that is located on SAP BTP from local applications.

To install the MongoDB database, execute the following steps:

Procedure

1. Download the MongoDB database from https://www.mongodb.com/download-center .
2. Unpack the file to a local directory (for example, `c:\mongodb`).
3. Create an empty directory (for example, `c:\mongodb_data`).
4. To start the MongoDB server, execute the following steps:
   a. Open a command prompt.
   b. Switch to the MongoDB `bin` directory (for example, `c:\mongodb\bin`).
   c. Enter the following command: `mongod --dbpath C:\mongodb_data`
5. In your Web browser, navigate to **`http://localhost:27017`**.

   If your setup is correct, you see a text message starting with "`You are trying to access MongoDB on the native driver port. …`"

Related Information

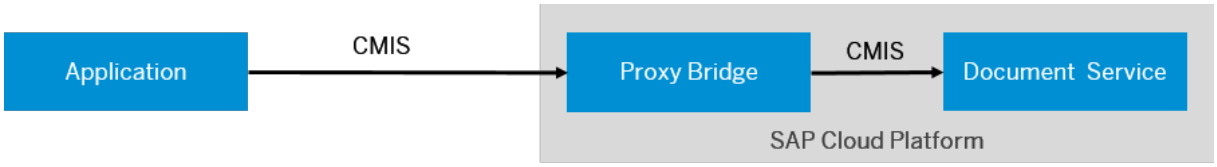
Create Sample Applications (Java) [page 557]

1.5.1.4 Access from External Applications

Overview

The services on SAP BTP can be consumed by applications that are deployed on SAP BTP but not from external applications. There are cases, however, where applications want to access content in the cloud but cannot be deployed in the cloud.

The figure below describes a mechanism with which this scenario can be supported and is followed by an explanation:



This can be addressed by deploying an application on SAP BTP that accepts incoming requests from the Internet and forwards them to the document service. We refer to this type of application as a proxy bridge. The proxy bridge is deployed on SAP BTP and runs in a subaccount using the common SAP BTP patterns. The proxy bridge is responsible for user authentication. The resources consumed in the document service are billed to the SAP BTP subaccount that deployed this application.

Related Information

Build a Proxy Bridge [page 562]

1.5.1.4.1 Build a Proxy Bridge

Context

All the standard mechanisms of the document service apply. The SAP BTP SDK provides a base class (a Java servlet) that provides the proxy functionality out-of-the-box. This can easily be extended to customize its behavior. The proxy bridge performs a 1:1 mapping from source CMIS calls to target CMIS calls. CMIS bindings can be enabled or disabled. Further modifications of the incoming requests, such as allowing only certain

operations or modifying parameters, are not supported. The Apache OpenCMIS project contains a bridge module that supports advanced scenarios of this type.

To experience the best performance and to benefit from the consistency model described in Consistency Model (Java) [page 557], ensure that cookies are enabled for client applications that connect to the proxy bridge. This is the default setting for HTML5 apps. Only if cookies are enabled, will your subsequent requests be dispatched to the same processing nodes, which is a prerequisite for the consistency model mentioned earlier.

The proxy bridge allows you to use standard CMIS clients to connect to the document service of SAP BTP. An example is the Apache Chemistry Workbench, which can be useful for development and testing.

> ⚠ Caution
>
> Note that the proxy bridge opens your repository to the public Internet and should always be secured appropriately.

> i Note
>
> For historic reasons, `ecm` is used to refer to the document service in the coding and the coding samples.

Procedure

1. Create an SAP BTP application as described in Using Java EE Web Profile Runtimes.
2. Create a `web.xml` file and a servlet class.
3. Derive your servlet from the class `com.sap.ecm.api.AbstractCmisProxyServlet`.
4. Add a servlet mapping to your `web.xml` file using a URL pattern that contains a wildcard. See the following example.

   > Example
   >
   ```
   <servlet>
     <servlet-name>cmisproxy</servlet-name>
     <servlet-class>my.app.CMISProxyServlet</servlet-class>
   </servlet>
   <servlet-mapping>
     <servlet-name>cmisproxy</servlet-name>
     <url-pattern>/cmis/*</url-pattern>
   </servlet-mapping>
   ```

   You can use prefixes other than `/cmis` and you can add more servlets in accordance with your needs. The URL pattern for your servlet derived from the class `AbstractCmisProxyServlet` must contain a `/*` suffix.
5. Override the two abstract methods provided by the `AbstractCmisProxyServlet` class: `getRepositoryUniqueName()` and `getRepositoryKey()`.

   These methods return a string containing the unique name and the secret key of the repository to be accessed. You can override a third method `getDestinationName()`, which also returns a string. If this method is overridden, it should return the name of a destination deployed for this application to connect to

the service. This is useful if a service user is used, for example. Ensure that there is a valid custom destination.

6. If you override the `getServletConfig()` method ensure that you call the superclass in your method.

   Do **not** override the following methods:

   - `service()`
   - `doGet()`
   - `doPost()`
   - and so on

7. Optionally, you can restrict the proxy bridge to restrict the exposed bindings by overriding one or more of the following methods:

   - `supportAtomPubBinding()`
   - `supportBrowserBinding()`

   At least one of the methods must return true.

8. Add the following code snippet to your `web.xml` and assign the role `EcmDeveloper` to the users in your subaccount who require external access to the repository.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Proxy</web-resource-name>
    <url-pattern>/cmis/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>EcmDeveloper</role-name>
  </auth-constraint>
</security-constraint>
```

   In some cases it might be useful to grant public access for reading content but not for modifying, creating or deleting it. For example, a Web content management application might embed pictures into a public Web site but store them in the document service. For a scenario of this type, override the method `readOnlyMode()` so that it returns true. This means that only read requests are forwarded to the repository and all other requests are rejected. The read-only mode only works with the JSON binding. The other bindings are disabled in this case.

   > **i Note**
   >
   > If you need finer control or dynamic permissions you can override the `requireAuthentication()` and `authenticate()` methods in the `AbstractCmisProxyServlet`.

9. Optionally, you can override two more methods to customize timeout values for reading and connecting: `getConnectTimeout()` and `getReadTimeout()`.

   It should only be necessary to use these methods if **frequent** timeout errors occur.

   The following code is an example of a proxy servlet.

```
package my.app;
import com.sap.ecm.api.AbstractCmisProxyServlet;
public class CMISProxyServlet extends AbstractCmisProxyServlet {
   @Override
   protected String getRepositoryUniqueName() {
       return "MySampleRepository";
   }
   @Override
```

```
   //For applications in production, use a secure location to store the
secret key.
   protected String getRepositoryKey() {
       return "abcdef0123456789";
   }
}
```

10. To access the proxy bridge from an external application you need the correct URL.

    The URL is composed of the following elements:

    - The URL of your deployed application (displayed in the cockpit)
    - The name of your Web application
    - The path configured in `web.xml` in the servlet mapping of the proxy bridge (in the example below: `/cmis`)
    - An extension for the CMIS binding type (`/atom` for AtomPub or `/json` for browser binding)

    ### Example

    Your proxy bridge application is deployed as `cmisproxy.war`. The cockpit shows the following URL for your app: `https://cmisproxysap.hana.ondemand.com/cmisproxy` and the `web.xml` is as shown above. Then the URLs is as follows:

    - CMIS 1.1:
      AtomPub: `https://cmisproxysap.hana.ondemand.com/cmisproxy/cmis/1.1/atom`
      Browser: `https://cmisproxysap.hana.ondemand.com/cmisproxy/cmis/json`
    - CMIS 1.0:
      AtomPub: `https://cmisproxysap.hana.ondemand.com/cmisproxy/cmis/atom`
      Browser: (not available)

    These URLs can be passed to the CMIS Workbench from Apache Chemistry, for example.

    The workbench requires basic authentication. Please add the following code to your `web.xml`:

    #### Sample Code

    ```
    <login-config>
    <auth-method>BASIC</auth-method>
    </login-config>
    ```

    ### Example

    A full example that can be deployed consists of two files: a `web.xml` and a servlet class. This example only exposes the CMIS browser binding (JSON) using the prefix `/cmis` in the URL.

    #### Sample Code

    `web.xml`

    ```
    <?xml version="1.0" encoding="UTF-8"?>
    <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://
    java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
    app_3_0.xsd" id="WebApp_ID" version="3.0">
      <display-name>cmisproxy</display-name>
    <servlet>
      <servlet-name>cmisproxy</servlet-name>
    ```

```
  <servlet-class>com.sap.ecm.proxy.CMISProxyServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>cmisproxy</servlet-name>
  <url-pattern>/cmis/*</url-pattern>
</servlet-mapping>
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Proxy</web-resource-name>
    <url-pattern>/cmis/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>EcmDeveloper</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
    <auth-method>BASIC</auth-method>
</login-config>
</web-app>
```

Sample Code

Servlet

```
package my.app;
import com.sap.ecm.api.AbstractCmisProxyServlet;
public class CMISProxyServlet extends AbstractCmisProxyServlet {
    private static final long serialVersionUID = 1L;
    @Override
    protected boolean supportAtomPubBinding() {
        return false;
    }
    @Override
    protected boolean supportBrowserBinding() {
        return true;
    }

    public CMISProxyServlet() {
        super();
    }
    @Override
    protected String getRepositoryUniqueName() {
        return "MySampleRepository";
    }
    @Override
    // For applications in production, use a secure location to store the
secret key.
    protected String getRepositoryKey() {
        return "abcdef0123456789";
    }
}
```

1.5.1.4.2 Access the Document Service from an HTML5 Application

Procedure

1. Create a repository in the subaccount area of cockpit.

   For more information, see Create Repositories (Cockpit) [page 599].
2. Create, deploy, and start a simple Java application as CMIS proxy bridge to get external access to the repository.

   Your repository should never be available to the public. In the example, basic authentication and the role *EcmDeveloper* are required (see security pages). Assign this role to the users or groups who should be able to access the subaccount area of cockpit.

   For more information, see Build a Proxy Bridge [page 562].
3. (Optional) Use CMIS workbench to test proxy bridge.

   For more information, see Build a Proxy Bridge [page 562].
4. Create a destination in the subaccount area of the cockpit to the proxy bridge using the following data:

Field	Value
Description	**`Connection to Proxy Bridge App`**
Type	**`HTTP`**
Authentication	*Basic Authentication*
Name	**`documentservice`**
CloudConnectorVersion	**`2`**
ProxyType	**`Internet`**
URL	**`https://cmisproxy<subaccount_ID>.hana.ondemand.com/cmisproxy/cmis/json`**
User	**`<your user ID>`**

   For more information, see Create HTTP Destinations [page 80].
5. Create an HTML5 application accessing the document service and open it in the Web IDE. Then create an `index.html` file with the following contents:

Example

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Use CMIS from HTML5 Application</title>
<script type="text/javascript">
    function setFilename() {
        var thefile = document.getElementById('filename').split('\
\').pop();
        document.getElementById("cmisname").value = thefile.value;
    }
    function getChildren() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                var children = obj = JSON.parse(this.responseText);
                var str = "<ul>";
                var repoUrl = "/cmis/<repo-ID>/root/"
                for (var i = 0; i <children.objects.length; i++) {
                    if
(children.objects[i].object.properties["cmis:baseTypeId"].value ==
'cmis:folder') {
                        str += '<li>'
                                +
children.objects[i].object.properties["cmis:name"].value
                                + ' (folder)</li>';
                    } else {
                        var name =
children.objects[i].object.properties["cmis:name"].value;
                        str += '<li><a href="' + repoUrl + name + '">' +
name
                                + '</a></li>';
                    }
                }
                str += "</ul>";
                document.getElementById("listchildren").innerHTML = str;
            }
        };
        xhttp.open("GET",
                "/cmis/<repo-id>/root?cmisselector=children",
                true);
        xhttp.send();
    }
</script>
</head>
<body>
    <h1>Document Service from HTML App</h1>
    <p>
        <a href="https://<your-proxy-url>"> Link
            to</a> Repository Info from Repository (requires
authentication) <br />
        <a href="/cmis/<repo-ID>?cmisselector=repositoryInfo">Link
            using destination to </a> Repository Info from Repository
    </p>
    <hr/>
    Upload a file to the document service (browse, then press upload)
    <form action="/cmis/<repo-ID>/root"
        enctype="multipart/form-data" method="post">
        <p>
            Please specify a file:<br> <input type="file" id="filename"
                onchange="setFilename()" name="datafile" size="40">
        </p>
        <div>
            <input type="submit" value="Upload">
```

```
                <input name="cmisaction" type="hidden"
value="createDocument"/>
                <input name="propertyId[0]" type="hidden"
value="cmis:objectTypeId"/>
                <input name="propertyValue[0]" type="hidden"
value="cmis:document"/>
                <input name="propertyId[1]" type="hidden"
value="cmis:name"/>
                <input name="propertyValue[1]" type="hidden"
id="cmisname"/>
        </div>
    </form>
    <hr/>
    List all objects in the root folder of your repository (press button
    after uploding to refresh)
    <br/>
    <input type="button" onclick="getChildren()" value="List">
    <div id="listchildren"></div>
</body>
</html>
```

For more information, see Create an HTML5 Application [page 1126], Create a Project [page 1121], and Edit the HTML5 Application [page 1122].

a. Open the URL of the proxy bridge from the previous step in a browser, copy the repository ID, for example, `8d1c2718db5a2fc0d7242585`, from the response.

   Example: `https://cmisproxyd058463sapdev.int.sap.hana.ondemand.com/cmisproxy/cmis/json`

   Example

   ```
   {
   "8d1c2718db5a2fc0d7242585": {
   "repositoryId": "8d1c2718db5a2fc0d7242585",
   "repositoryName": "Sample Repository",
   "repositoryDescription": "Sample repository for external access",
   "vendorName": "SAP AG",
   "productName": "SAP Cloud Platform, document service",
   "productVersion": "1.0",
   "rootFolderId": "8d1c2718db5a2fc0d7242585",
   "capabilities": {
   …
   ```

b. In your `index.html`, replace all occurrences of `<repo-id>` with the extracted repository ID and all occurrences of `<your-proxy-url>` with the URL of the proxy bridge application.

c. Create a `neo-app.json` file in the root of your project directory with the following contents:

   Example

   ```
   {
     "welcomeFile": "/index.html",
     "routes": [
       {
         "path": "/cmis",
         "target": {
           "type": "destination",
           "name": "documentservice"
         },
         "description": "CMIS Connection Document Service"
       }
     ],
   ```

```
  "sendWelcomeFileRedirect": true
}
```

This handles all URLs starting with `/cmis` to the path specified in the destination named “documentservice”.

d. Commit your files in Git, create a new version, and activate the version.

For more information, see Create a Version [page 1127] and Activate a Version [page 1128].

1.5.1.5 Advanced Concepts

The following sections describe the advanced concepts of the SAP Document service.

- Handling CMIS Metadata [page 570]
- ACLs in the Document Service [page 588]

1.5.1.5.1 Handling CMIS Metadata

One benefit of Content Management Interoperability Services (CMIS) as compared to a file system is the extended handling of metadata.

You can use metadata to structure content and make it easier to find documents in a repository, even if it contains millions of documents. In the CMIS domain model, metadata is structured using types. A type contains the set of allowed or required properties, for example, an `Invoice` type that has the `InvoiceNo` and `CustomerNo` properties.

The CMIS Type System

A type is described in a type definition and contains a list of property definitions. CMIS has a set of predefined types and predefined properties. Custom-specific types and additional custom properties can extend the predefined types. When a type is created, it is derived from a parent type and extends the set of the parent properties. In this way, a hierarchy of types is built. The base types do not have parents. Base types are defined in the CMIS specification. The most important base types are `cmis:document` and `cmis:folder`.

Predefined properties contain metadata that is usually available in the existing repositories. These are, for example, `cmis:name`, `cmis:createdBy`, `cmis:modifiedBy`, `cmis:createdAt`, and `cmis:modifiedAt`. They contain the name of the author, the creation date, and the date of the last modification. Some properties are type-specific, for example, a folder has a parent folder and a document has a property for content length.

Each property has a data format (String, Integer, Date, Decimal, ID, and so on) and can define additional constraints, such as:

- Required (must have a value)
- Read-only (cannot be updated)

- Value range (minimum value, maximum value)
- Value list (value must be one of a fixed list of values)

A property is either single-valued or multi-valued.

Each object stored in a CMIS repository has a type and a set of properties. Types and properties provide the mechanism used to find objects with CMIS queries.

Related Information

http://chemistry.apache.org/
http://chemistry.apache.org/java/developing/guide.html
http://chemistry.apache.org/java/0.9.0/maven/apidocs/
http://chemistry.apache.org/java/examples/index.html
http://docs.oasis-open.org/cmis/CMIS/v1.1
http://docs.oracle.com/javase/6/docs/api/java/security/KeyStore.html

1.5.1.5.1.1 Metadata in the Document Service

The document store on SAP BTP supports the `cmis:document` and `cmis:folder` types. It also has a built-in subtype for versioned documents. The types can be investigated using the Apache CMIS workbench.

In addition to the standard CMIS properties, the document service of SAP BTP supports additional SAP properties. The most important ones are:

- `sap:owner` (the owner of a document)
  `sap:owner` is used for ACLs (access control).
- `sap:tags`
  `sap:tags` is a multi-valued attribute used for tagging the content.

Related Information

http://chemistry.apache.org/java/download.html
http://docs.oasis-open.org/cmis/CMIS/v1.1

1.5.1.5.1.2 Create Metadata

You can pass metadata when documents are created.

Context

The CMIS client API uses a map to pass properties. The key of the map is the property ID and the value is the actual value to be passed. The `cmis:name` and `cmis:objectTypeId` properties are mandatory.

Procedure

1. Use a name that is unique within the folder and a type ID that is a valid type from the repository.
2. Run the sample code.

```
// properties
Map<String, Object> properties = new HashMap<String, Object>();
properties.put(PropertyIds.OBJECT_TYPE_ID, "cmis:document");
properties.put(PropertyIds.NAME, "Document-1");
// content
byte[] content = "Hello World!".getBytes();
InputStream stream = new ByteArrayInputStream(content);
ContentStream contentStream = new ContentStreamImpl(name,
BigInteger.valueOf(content.length), "text/plain", stream);
// create a document
Folder root = session.getRootFolder();
Document newDoc = folder.createDocument(properties, contentStream,
VersioningState.NONE
```

Results

You can inspect the document in the CMIS workbench. You can see that various other properties have been set by the system, such as the ID, the creation date, and the creating user.

1.5.1.5.1.3 Update Properties

Context

This procedure focuses on the use of the `sap:tags` property to mark the document. This is a multi-value attribute, so you can assign more than one tag to it.

Procedure

1. To assign the `Hello` and `Tutorial` tags to the document, use the following code:

```
List<String> tags = Arrays.asList("Hello", "Tutorial");
Map<String, Object> properties = new HashMap<String, Object>();
properties.put("sap:tags", tags);
doc.updateProperties(properties);
```

2. To display the property, refresh the document in the CMIS workbench.

   The following property is displayed:

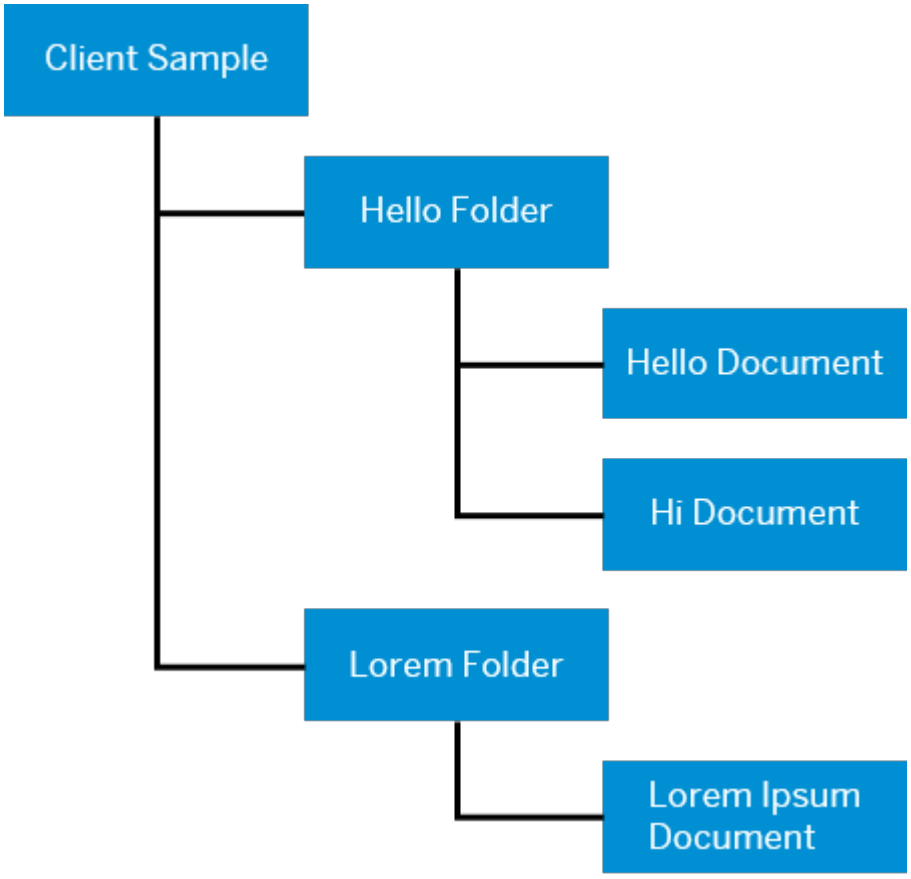
Name	ID	Type	Value
`sap:tags`	`sap:tags`	string	Hello Tutorial

1.5.1.5.1.4 Query Properties

This section gives a very brief introduction to querying. The OpenCMIS Client API is a Java client-side library with many capabilities, for example, paging results. For more information, consult the OpenCMIS Javadoc and the examples on the Apache Chemistry Web site.

Context

The following procedure focuses on a use case where you have created a second folder and some more documents. The repository then looks like this:



The `Hello Document` and `Hi Document` documents have the tags `Hello` and `Tutorial`, the `Loren Ipsum` document has no tags.

Procedure

1. Use the CMIS query to search documents in the system based on their properties.

> **i Note**
>
> The CMIS query language CMISSQL is similar to SQL.

```
SELECT cmis:name, cmis:objectId, cmis:createdBy, sap:owner FROM cmis:document
WHERE cmis:createdBy='john'
```

This query returns the following result set:

cmis:createdBy	cmis:name	sap:owner	cmis:objectId
john	Lorem Ipsum Document	john	<ID>
john	Hi Document	john	<ID>
john	Hello Document	john	<ID>

2. Query all documents with the tag `Hello`.

```
SELECT cmis:name, cmis:objectId, cmis:createdBy, sap:tags, sap:owner FROM
cmis:document WHERE ANY sap:tags IN ('Hello')
```

> **i Note**
>
> In this case, the workbench displays only the first value of multivalued properties.

cmis:createdBy	cmis:name	sap:owner	sap:tags	cmis:objectId
john	Hello Document	john	Hello Tutorial	<ID>
john	Hi Document	john	Hello Tutorial	<ID>

3. Execute the query with the following code:

```
String query = "SELECT cmis:name, cmis:objectId, cmis:createdBy, " +
     "sap:tags, sap:owner FROM cmis:document WHERE ANY sap:tags " +
"IN ('Hello')";
ItemIterable<QueryResult> results = session.query(query, false););
System.out.println("Name | Object-Id  | Author | Tags");
System.out.println("---------------------------------");
for (QueryResult result : results) {
    String name = result.getPropertyValueByQueryName("cmis:name");
    String id =result.getPropertyValueByQueryName("cmis:objectId");
    String author =
        result.getPropertyValueByQueryName("cmis:objectId");
    List<String> tags =
        result.getPropertyMultivalueByQueryName("sap:tags");
    System.out.println(name + " | " + id + " | " + author + " | "
        + tags);
}
```

This query produces the following output:

```
Name | Object-Id | Author | Tags --------------------------------- Hello-
Document | 7L8S0XYG9dh7O1gGgkiA9gWl3gSIzYkYpYds6vnxA-M | john | [Hello,
Tutorial] Hi-Document | 3ovtYi1sqWyUmXW3-zGg30OT-e2U12qiD_o-kf595YA | john |
[Hello, Tutorial]
```

Related Information

http://chemistry.apache.org/java/0.13.0/maven/apidocs/

http://chemistry.apache.org/java/examples/index.html

1.5.1.5.1.5 Mutability of Object Types

For the SAP Document service, you can create new object types or you can remove those new object types again in accordance with the CMIS standard.

Context

In CMIS, every object, for example a document or a folder, has an object type. The object type defines the basic settings of an object of that type. For example, the `cmis:document` object type defines that objects of that type are searchable.

Furthermore, the object type defines the properties that can be set for an object of that type, for example, an object of type `cmis:document` has a mandatory `cmis:name` property that must be a string. Therefore, every object of type `cmis:document` needs a name. Otherwise, the object is not valid and the repository rejects it.

In CMIS, types are organized hierarchically. The most important (predefined) base types are:

- `cmis:document` for all file-like objects
- `cmis:folder` for folder-like objects
- `cmis:secondary` for secondary types

CMIS allows you to define additional types provided that each type is a descendant of one of the predefined base types. In this type hierarchy, a type inherits all property definitions of its parent type. CMIS 1.1 allows type hierarchy modifications (see the OASIS page) by providing methods for the creation, the modification, and the removal of object types. Currently, the document service only supports the creation and removal of types. This allows a developer to define new types as subtypes of existing types. The new types might possess other properties in addition to all of the automatically inherited property definitions of the parent type. Creating objects of that type allows you to assign values for these new properties to the object. Remember to also set the values for the inherited properties as appropriate.

The following example shows how to create a new document type that possesses one additional property for storing the summary of a document. The developer must implement the `MyDocumentTypeDefinition` and `MyStringPropertyDefinition` classes. Example implementations for these classes as well as for the interfaces (`FolderTypeDefinition`, `SecondaryTypeDefinition`, `PropertyBooleanDefinition`, `PropertyDecimalDefinition`, and so on) are described in the following topics.

Example

```
import java.util.HashMap;
import java.util.Map;
import org.apache.chemistry.opencmis.client.api.ObjectType;
import org.apache.chemistry.opencmis.client.api.Session;
import org.apache.chemistry.opencmis.commons.definitions.PropertyDefinition;
import org.apache.chemistry.opencmis.commons.enums.BaseTypeId;
import org.apache.chemistry.opencmis.commons.enums.Cardinality;
import org.apache.chemistry.opencmis.commons.enums.ContentStreamAllowed;
import org.apache.chemistry.opencmis.commons.enums.Updatability;
import
org.apache.chemistry.opencmis.commons.exceptions.CmisObjectNotFoundException;
import org.apache.chemistry.opencmis.commons.exceptions.CmisRuntimeException;
// specify type attributes
String idAndQueryName = "test:docWithSummary";
String description = "Doc with Summary";
String displayName = "Document with Summary";
String localName = "some local name";
String localNamespace = "some local name space";
String parentTypeId = BaseTypeId.CMIS_DOCUMENT.value();
Boolean isCreatable = true;
Boolean includedInSupertypeQuery = true;
Boolean queryable = true;
ContentStreamAllowed contentStreamAllowed = ContentStreamAllowed.ALLOWED;
Boolean versionable = false;
// specify property definitions
Map<String, PropertyDefinition<?>> propertyDefinitions
  = new HashMap<String, PropertyDefinition<?>>();
MyStringPropertyDefinition summaryPropertyDefinitions
  = createSummaryPropertyDefinitions();
propertyDefinitions.put(summaryPropertyDefinitions.getId(),
summaryPropertyDefinitions);
// build object type
MyDocumentTypeDefinition docTypeDefinition
  = new MyDocumentTypeDefinition(idAndQueryName, description, displayName,
    localName, localNamespace, parentTypeId, isCreatable,
    includedInSupertypeQuery, queryable, contentStreamAllowed,
    versionable, propertyDefinitions);
// add type to repository
ecmSession.createType(docTypeDefinition);
// create document of new type
ecmSession.clear();
Map<String, String> newDocProps = new HashMap<String, String>();
newDocProps.put(PropertyIds.OBJECT_TYPE_ID, docTypeDefinition.getId());
newDocProps.put(PropertyIds.NAME, "testDocWithNewType");
newDocProps.put("test:summary", "This is a document with a summary property");

Folder root = ecmSession.getRootFolder();
root.createDocument(newDocProps, null, null);
private static MyStringPropertyDefinition createSummaryPropertyDefinitions() {
  String idAndQueryName = "test:summary";
  Cardinality cardinality = Cardinality.SINGLE;
  String description = "this is a summary";
  String displayName = "Summary";
  String localName = "some local name";
  String localNameSpace = "some local name space";

  Updatability updatability = Updatability.READWRITE;
  Boolean orderable = false;
  Boolean queryable = false;
  MyStringPropertyDefinition summaryPropDef
    = new  MyStringPropertyDefinition(idAndQueryName,
      cardinality, description, displayName, localName, localNameSpace,
      updatability, orderable, queryable);
  return summaryPropDef;
}
```

Restrictions for Creating New Object Types

- There is a limit of 120 characters for the IDs of type names.
- You can only create types with a `cmis:document`, `cmis:folder`, or `cmis:secondary` base type.
- The ID and the query name must be identical and meet the following rules:
  - They must match the regular Java expression "`[a-zA-Z][a-zA-Z0-9_:]*`".
  - Their names must not start with `cmis:`, `sap`, or `s:` in any combination of uppercase and lowercase letters, for example, `cMis:` is also not allowed.

Restrictions for Property Definitions of New Object Types

- The ID and the query name must be identical and meet the following rules:
  - They must match the regular Java expression "`[a-zA-Z][a-zA-Z0-9_:]*`".
  - Their names must not start with `cmis:`, `sap`, or `s:` in any combination of uppercase and lowercase letters, for example, `cMis:` is also not allowed.
- If the base type of the new object type is `cmis:secondary`, no other type definition may already contain a property definition with the same ID or query name.
- If the base type of the new object type is not `cmis:secondary` and another type definition already contains a property definition with the same ID or query name, this property definition must be identical to the one of the new type.
- You cannot specify default values or choices.

Deleting New Object Types

To delete a new object type, you can use the following code snippet: `ecmSession.deleteType(typeId);`

You can only delete an object type if it is no longer used by any documents or folders in the repository.

Updating Object Types

Updating an object type is not supported.

Related Information

OASIS page

1.5.1.5.1.5.1 Implementation Examples for Type and Property Definition Classes

Example

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.apache.chemistry.opencmis.commons.data.CmisExtensionElement;
import org.apache.chemistry.opencmis.commons.definitions.PropertyDefinition;
import org.apache.chemistry.opencmis.commons.definitions.TypeDefinition;
import org.apache.chemistry.opencmis.commons.definitions.TypeMutability;
import org.apache.chemistry.opencmis.commons.enums.BaseTypeId;
public abstract class MyTypeDefinition implements TypeDefinition {
  private String description = null;
  private String displayName = null;
  private String idAndQueryName = null;
  private String localName = null;
  private String localNamespace = null;
  private String parentTypeId = null;
  private Boolean isCreatable = null;
  private Boolean includedInSupertypeQuery = null;
  private Boolean queryable = null;
  private Map<String, PropertyDefinition<?>> propertyDefinitions
    = new HashMap<String, PropertyDefinition<?>>();
  public MyTypeDefinition(String idAndQueryName, String description,
    String displayName, String localName, String localNamespace,
    String parentTypeId, Boolean isCreatable,
    Boolean includedInSupertypeQuery, Boolean queryable,
    Map<String, PropertyDefinition<?>> propertyDefinitions) {
    this.description = description;
    this.displayName = displayName;
    this.idAndQueryName = idAndQueryName;
    this.localName = localName;
    this.localNamespace = localNamespace;
    this.parentTypeId = parentTypeId;
    this.isCreatable = isCreatable;
    this.includedInSupertypeQuery = includedInSupertypeQuery;
    this.queryable = queryable;
    if (propertyDefinitions != null) {
      this.propertyDefinitions = propertyDefinitions;
    }
  }
  @Override
  abstract public BaseTypeId getBaseTypeId();
  @Override
  public String getDescription() {
    return description;
  }
  @Override
  public String getDisplayName() {
    return displayName;
  }
  @Override
  public String getId() {
    return idAndQueryName;
  }
  @Override
  public String getLocalName() {
```

```
    return localName;
  }
  @Override
  public String getLocalNamespace() {
    return localNamespace;
  }
  @Override
  public String getParentTypeId() {
    return parentTypeId;
  }
  @Override
  public Map<String, PropertyDefinition<?>> getPropertyDefinitions() {
    return propertyDefinitions;
  }
  @Override
  public String getQueryName() {
    return idAndQueryName;
  }
  @Override
  public Boolean isCreatable() {
    return isCreatable;
  }
  @Override
  public Boolean isIncludedInSupertypeQuery() {
    return includedInSupertypeQuery;
  }
  @Override
  public Boolean isQueryable() {
    return queryable;
  }
  // methods with static content
  @Override
  public TypeMutability getTypeMutability() {
    return new MyTypeMutability();
  }
  @Override
  public Boolean isControllableAcl() {
    return true;
  }
  @Override
  public Boolean isControllablePolicy() {
    return false;
  }
  @Override
  public Boolean isFileable() {
    return true;
  }
  @Override
  public Boolean isFulltextIndexed() {
    return false;
  }
  @Override
  public List<CmisExtensionElement> getExtensions() {
    return null;
  }
  @Override
  public void setExtensions(List<CmisExtensionElement> extension) {
  }
}
```

1.5.1.5.1.5.1.1 MyTypeMutability Class

```
import java.util.List;
import org.apache.chemistry.opencmis.commons.data.CmisExtensionElement;
import org.apache.chemistry.opencmis.commons.definitions.TypeMutability;
public class MyTypeMutability implements TypeMutability {
  @Override
  public List<CmisExtensionElement> getExtensions() {
    return null;
  }
  @Override
  public void setExtensions(List<CmisExtensionElement> arg0) {
  }
  @Override
  public Boolean canCreate() {
    return true;
  }
  @Override
  public Boolean canDelete() {
    return true;
  }
  @Override
  public Boolean canUpdate() {
    return false;
  }
}
```

1.5.1.5.1.5.1.2 MyDocumentTypeDefinition Class

```
import java.util.Map;
import org.apache.chemistry.opencmis.commons.definitions.DocumentTypeDefinition;
import org.apache.chemistry.opencmis.commons.definitions.PropertyDefinition;
import org.apache.chemistry.opencmis.commons.enums.BaseTypeId;
import org.apache.chemistry.opencmis.commons.enums.ContentStreamAllowed;
public class MyDocumentTypeDefinition extends MyTypeDefinition implements
DocumentTypeDefinition {
  private ContentStreamAllowed contentStreamAllowed = null;
  private Boolean versionable = null;
  public MyDocumentTypeDefinition(String idAndQueryName, String description,
    String displayName, String localName, String localNamespace,
    String parentTypeId, Boolean isCreatable,
    Boolean includedInSupertypeQuery, Boolean queryable,
    ContentStreamAllowed contentStreamAllowed, Boolean versionable,
    Map<String, PropertyDefinition<?>> propertyDefinitions) {
    super(idAndQueryName, description, displayName, localName, localNamespace,
      parentTypeId, isCreatable, includedInSupertypeQuery, queryable,
propertyDefinitions);
    this.contentStreamAllowed = contentStreamAllowed;
    this.versionable = versionable;
  }
  @Override
  public BaseTypeId getBaseTypeId() {
    return BaseTypeId.CMIS_DOCUMENT;
  }
  @Override
  public ContentStreamAllowed getContentStreamAllowed() {
    return contentStreamAllowed;
  }
  @Override
```

```
  public Boolean isVersionable() {
    return versionable;
  }
}
```

1.5.1.5.1.5.1.3 MyFolderTypeDefinition Class

```
import java.util.Map;
import org.apache.chemistry.opencmis.commons.definitions.FolderTypeDefinition;
import org.apache.chemistry.opencmis.commons.definitions.PropertyDefinition;
import org.apache.chemistry.opencmis.commons.enums.BaseTypeId;
public class MyFolderTypeDefinition extends MyTypeDefinition implements
FolderTypeDefinition {
  public MyFolderTypeDefinition(String idAndQueryName, String description,
    String displayName, String localName, String localNamespace,
    String parentTypeId, Boolean isCreatable,
    Boolean includedInSupertypeQuery, Boolean queryable,
    Map<String, PropertyDefinition<?>> propertyDefinitions) {
    super(idAndQueryName, description, displayName, localName, localNamespace,
      parentTypeId, isCreatable, includedInSupertypeQuery, queryable,
      propertyDefinitions);
  }
  @Override
  public BaseTypeId getBaseTypeId() {
    return BaseTypeId.CMIS_FOLDER;
  }
}
```

1.5.1.5.1.5.1.4 MySecondaryTypeDefinition Class

```
import java.util.Map;
import org.apache.chemistry.opencmis.commons.definitions.FolderTypeDefinition;
import org.apache.chemistry.opencmis.commons.definitions.PropertyDefinition;
import org.apache.chemistry.opencmis.commons.enums.BaseTypeId;
public class MySecondaryTypeDefinition extends MyTypeDefinition implements
FolderTypeDefinition {
  public MySecondaryTypeDefinition(String idAndQueryName, String description,
    String displayName, String localName, String localNamespace,
    String parentTypeId, Boolean isCreatable,
    Boolean includedInSupertypeQuery, Boolean queryable,
    Map<String, PropertyDefinition<?>> propertyDefinitions) {
    super(idAndQueryName, description, displayName, localName, localNamespace,
      parentTypeId, isCreatable, includedInSupertypeQuery, queryable,
      propertyDefinitions);
  }
  @Override
  public BaseTypeId getBaseTypeId() {
    return BaseTypeId.CMIS_SECONDARY;
  }
}
```

1.5.1.5.1.5.1.5 MyPropertyDefinition Class

```
import java.util.List;
import org.apache.chemistry.opencmis.commons.data.CmisExtensionElement;
import org.apache.chemistry.opencmis.commons.definitions.Choice;
import org.apache.chemistry.opencmis.commons.definitions.PropertyDefinition;
import org.apache.chemistry.opencmis.commons.enums.Cardinality;
import org.apache.chemistry.opencmis.commons.enums.PropertyType;
import org.apache.chemistry.opencmis.commons.enums.Updatability;
abstract public class MyPropertyDefinition<T> implements PropertyDefinition<T> {
  private String idAndQueryName = null;
  private Cardinality cardinality = null;
  private String description = null;
  private String displayName = null;
  private String localName = null;
  private String localNameSpace = null;
  private Updatability updatability = null;
  private Boolean orderable = null;
  private Boolean queryable = null;
  public MyPropertyDefinition(String idAndQueryName, Cardinality cardinality,
    String description, String displayName, String localName,
    String localNameSpace, Updatability updatability,
    Boolean orderable, Boolean queryable) {
    super();
    this.idAndQueryName = idAndQueryName;
    this.cardinality = cardinality;
    this.description = description;
    this.displayName = displayName;
    this.localName = localName;
    this.localNameSpace = localNameSpace;
    this.updatability = updatability;
    this.orderable = orderable;
    this.queryable = queryable;
  }
  @Override
  public String getId() {
    return idAndQueryName;
  }
  @Override
  public Cardinality getCardinality() {
    return cardinality;
  }
  @Override
  public String getDescription() {
    return description;
  }
  @Override
  public String getDisplayName() {
    return displayName;
  }
  @Override
  public String getLocalName() {
    return localName;
  }
  @Override
  public String getLocalNamespace() {
    return localNameSpace;
  }
  @Override
  abstract public PropertyType getPropertyType();
  @Override
  public String getQueryName() {
    return idAndQueryName;
  }
  @Override
  public Updatability getUpdatability() {
```

```
    return updatability;
  }
  @Override
  public Boolean isOrderable() {
    return orderable;
  }
  @Override
  public Boolean isQueryable() {
    return queryable;
  }
  // methods with static content
  @Override
  public List<Choice<T>> getChoices() {
    return null;
  }
  @Override
  public List<T> getDefaultValue() {
    return null;
  }
  @Override
  public Boolean isInherited() {
    return false;
  }
  @Override
  public Boolean isOpenChoice() {
    return true;
  }
  @Override
  public Boolean isRequired() {
    return false;
  }
  @Override
  public List<CmisExtensionElement> getExtensions() {
    return null;
  }
  @Override
  public void setExtensions(List<CmisExtensionElement> arg0) {
  }
}
```

1.5.1.5.1.5.1.6 MyBooleanPropertyDefinition Class

```
import
org.apache.chemistry.opencmis.commons.definitions.PropertyBooleanDefinition;
import org.apache.chemistry.opencmis.commons.enums.Cardinality;
import org.apache.chemistry.opencmis.commons.enums.PropertyType;
import org.apache.chemistry.opencmis.commons.enums.Updatability;
public class MyBooleanPropertyDefinition extends MyPropertyDefinition<Boolean>
implements PropertyBooleanDefinition {
  public MyBooleanPropertyDefinition(String idAndQueryName,
    Cardinality cardinality, String description, String displayName,
    String localName, String localNameSpace,
    Updatability updatability, Boolean orderable, Boolean queryable) {

  super(idAndQueryName, cardinality, description, displayName,
    localName, localNameSpace, updatability, orderable, queryable);
  }
  @Override
  public PropertyType getPropertyType() {
    return PropertyType.BOOLEAN;
  }
}
```

1.5.1.5.1.5.1.7 MyDateTimePropertyDefinition Class

```
import java.util.GregorianCalendar;
import
org.apache.chemistry.opencmis.commons.definitions.PropertyDateTimeDefinition;
import org.apache.chemistry.opencmis.commons.enums.Cardinality;
import org.apache.chemistry.opencmis.commons.enums.DateTimeResolution;
import org.apache.chemistry.opencmis.commons.enums.PropertyType;
import org.apache.chemistry.opencmis.commons.enums.Updatability;
public class MyDateTimePropertyDefinition extends
MyPropertyDefinition<GregorianCalendar> implements PropertyDateTimeDefinition {
  public MyDateTimePropertyDefinition(String idAndQueryName,
    Cardinality cardinality, String description, String displayName,
    String localName, String localNameSpace,
    Updatability updatability, Boolean orderable, Boolean queryable) {
    super(idAndQueryName, cardinality, description, displayName,
      localName, localNameSpace, updatability, orderable, queryable);
  }
  @Override
  public PropertyType getPropertyType() {
    return PropertyType.DATETIME;
  }
  @Override
  public DateTimeResolution getDateTimeResolution() {
    return DateTimeResolution.TIME;
  }
}
```

1.5.1.5.1.5.1.8 MyDecimalPropertyDefinition Class

```
import java.math.BigDecimal;
import
org.apache.chemistry.opencmis.commons.definitions.PropertyDecimalDefinition;
import org.apache.chemistry.opencmis.commons.enums.Cardinality;
import org.apache.chemistry.opencmis.commons.enums.DecimalPrecision;
import org.apache.chemistry.opencmis.commons.enums.PropertyType;
import org.apache.chemistry.opencmis.commons.enums.Updatability;
public class MyDecimalPropertyDefinition extends
MyPropertyDefinition<BigDecimal> implements
    PropertyDecimalDefinition {
  public MyDecimalPropertyDefinition(String idAndQueryName,
    Cardinality cardinality, String description, String displayName,
    String localName, String localNameSpace,
    Updatability updatability, Boolean orderable, Boolean queryable) {
    super(idAndQueryName, cardinality, description, displayName,
      localName, localNameSpace, updatability, orderable, queryable);
  }
  @Override
  public PropertyType getPropertyType() {
    return PropertyType.DECIMAL;
  }
  @Override
  public BigDecimal getMaxValue() {
    return null;
  }
```

```
  @Override
  public BigDecimal getMinValue() {
    return null;
  }
  @Override
  public DecimalPrecision getPrecision() {
    return DecimalPrecision.BITS64;
  }
}
```

1.5.1.5.1.5.1.9 MyHtmlPropertyDefinition Class

```
import org.apache.chemistry.opencmis.commons.definitions.PropertyHtmlDefinition;
import org.apache.chemistry.opencmis.commons.enums.Cardinality;
import org.apache.chemistry.opencmis.commons.enums.PropertyType;
import org.apache.chemistry.opencmis.commons.enums.Updatability;
public class MyHtmlPropertyDefinition extends MyPropertyDefinition<String>
implements PropertyHtmlDefinition {
  public MyHtmlPropertyDefinition(String idAndQueryName,
    Cardinality cardinality, String description, String displayName,
    String localName, String localNameSpace,
    Updatability updatability, Boolean orderable, Boolean queryable) {
    super(idAndQueryName, cardinality, description, displayName,
      localName, localNameSpace, updatability, orderable, queryable);
  }
  @Override
  public PropertyType getPropertyType() {
    return PropertyType.HTML;
  }
}
```

1.5.1.5.1.5.1.10 MyIdPropertyDefinition Class

```
import org.apache.chemistry.opencmis.commons.definitions.PropertyIdDefinition;
import org.apache.chemistry.opencmis.commons.enums.Cardinality;
import org.apache.chemistry.opencmis.commons.enums.PropertyType;
import org.apache.chemistry.opencmis.commons.enums.Updatability;
public class MyIdPropertyDefinition extends MyPropertyDefinition<String>
implements PropertyIdDefinition {
  public MyIdPropertyDefinition(String idAndQueryName,
    Cardinality cardinality, String description, String displayName,
    String localName, String localNameSpace,
    Updatability updatability, Boolean orderable, Boolean queryable) {
    super(idAndQueryName, cardinality, description, displayName,
      localName, localNameSpace, updatability, orderable, queryable);
  }
  @Override
  public PropertyType getPropertyType() {
    return PropertyType.ID;
  }
}
```

1.5.1.5.1.5.1.11 MyIntegerPropertyDefinition Class

```
import java.math.BigInteger;
import
org.apache.chemistry.opencmis.commons.definitions.PropertyIntegerDefinition;
import org.apache.chemistry.opencmis.commons.enums.Cardinality;
import org.apache.chemistry.opencmis.commons.enums.PropertyType;
import org.apache.chemistry.opencmis.commons.enums.Updatability;
public class MyIntegerPropertyDefinition extends
MyPropertyDefinition<BigInteger> implements PropertyIntegerDefinition {
  public MyIntegerPropertyDefinition(String idAndQueryName,
    Cardinality cardinality, String description, String displayName,
    String localName, String localNameSpace,
    Updatability updatability, Boolean orderable, Boolean queryable) {
    super(idAndQueryName, cardinality, description, displayName,
      localName, localNameSpace, updatability, orderable, queryable);
  }
  @Override
  public PropertyType getPropertyType() {
    return PropertyType.INTEGER;
  }
  @Override
  public BigInteger getMaxValue() {
    return null;
  }
  @Override
  public BigInteger getMinValue() {
    return null;
  }
}
```

1.5.1.5.1.5.1.12 MyStringPropertyDefinition Class

```
import java.math.BigInteger;
import
org.apache.chemistry.opencmis.commons.definitions.PropertyStringDefinition;
import org.apache.chemistry.opencmis.commons.enums.Cardinality;
import org.apache.chemistry.opencmis.commons.enums.PropertyType;
import org.apache.chemistry.opencmis.commons.enums.Updatability;
public class MyStringPropertyDefinition extends MyPropertyDefinition<String>
implements PropertyStringDefinition {
  public MyStringPropertyDefinition(String idAndQueryName,
    Cardinality cardinality, String description, String displayName,
    String localName, String localNameSpace,
    Updatability updatability, Boolean orderable, Boolean queryable) {
    super(idAndQueryName, cardinality, description, displayName,
      localName, localNameSpace, updatability, orderable, queryable);
  }
  @Override
  public PropertyType getPropertyType() {
    return PropertyType.STRING;
  }
  @Override
  public BigInteger getMaxLength() {
    return null;
  }
}
```

1.5.1.5.1.5.1.13 MyUriPropertyDefinition Class

```
import org.apache.chemistry.opencmis.commons.definitions.PropertyUriDefinition;
import org.apache.chemistry.opencmis.commons.enums.Cardinality;
import org.apache.chemistry.opencmis.commons.enums.PropertyType;
import org.apache.chemistry.opencmis.commons.enums.Updatability;
public class MyUriPropertyDefinition extends MyPropertyDefinition<String>
implements
    PropertyUriDefinition {
  public MyUriPropertyDefinition(String idAndQueryName,
    Cardinality cardinality, String description, String displayName,
    String localName, String localNameSpace,
    Updatability updatability, Boolean orderable, Boolean queryable) {
    super(idAndQueryName, cardinality, description, displayName,
      localName, localNameSpace, updatability, orderable, queryable);
  }
  @Override
  public PropertyType getPropertyType() {
    return PropertyType.URI;
  }
}
```

1.5.1.5.2 ACLs in the Document Service

The document service supports ACLs (Access Control Lists) consisting of ACEs (Access Control Entries) to control the access to documents and folders as described in the CMIS standard.

The document service supports the following permissions:

- `cmis:read`
  - Allows fetching an object (folder or document).
  - Allows reading the ACL, properties and the content of an object.
- `sap:file`
  - Includes all privileges of `cmis:read`.
  - Allows the creation of objects in a folder and to move an object.
- `cmis:write`
  - Includes all privileges of `sap:file`.
  - Allows modifying the properties and the content of an object.
  - Allows checking out of a versionable document.
- `sap:delete`
  - Includes all privileges of `cmis:write`.
  - Allows the deletion of an object.
  - Allows checking in and canceling check out of a private working copy.
- `cmis:all`
  - Includes all privileges of `sap:delete`.
  - Allows modifying the ACL of an object.

For a repository the initial settings for the root folder are:

- The ACL contains one ACE for the `{sap:builtin}everyone` principal with the `cmis:all` permission. With these settings, all principals have full control over the root folder.

- The owner property is set to `{sap:builtin}admin` (ownership is described below).

Initially, without specific ACL settings, all documents and folders possess an ACL with one ACE for the built-in principal `{sap:builtin}everyone` with the `cmis:all` permission that grants all users unrestricted access.

ACLs or ACEs are not inherited but explicitly stored at the particular objects. An empty ACL means that no principal has permission, except the owner of the object. The owner concept is described below in more detail.

Example

The example assumes that every user has full access to the folder. In the following, the access to a folder is restricted in such a way that User1 has full access and User2 has only read access.

```
Session session = ..;
Folder folder = ..;
String userIdOfUser1 = ..;
String userIdOfUser2 = ..;
// list of ACEs which should be added
List<Ace> addAcl = new ArrayList<Ace>();
// build and add ACE for user U1
List<String> permissionsUser1 = new ArrayList<String>();
    permissionsUser1.add("cmis:all");
Ace aceUser1 = session.getObjectFactory().createAce(userIdOfUser1,
        permissionsUser1);
addAcl.add(aceUser1);
// build and add ACE for user U2
List<String> permissionsUser2 = new ArrayList<String>();
    permissionsUser2.add("cmis:read");
Ace aceUser2 = session.getObjectFactory().createAce(userIdOfUser2,
    permissionsUser1);
addAcl.add(aceUser2);
// list of ACEs which should be removed
List<Ace> removeAcl = new ArrayList<Ace>();
// build and add ACE for user {sap:builtin}everyone
List<String> permissionsEveryone = new ArrayList<String>();
    permissionsEveryone.add("cmis:all");
Ace aceEveryone = session.getObjectFactory().createAce(
        "{sap:builtin}everyone", permissionsEveryone);
removeAcl.add(aceEveryone);

// add and remove the ACEs at the folder
folder.applyAcl(addAcl, removeAcl, AclPropagation.OBJECTONLY);
```

1.5.1.5.2.1 Detailed Method Description

The following methods for modifying ACLs (Access Control Lists) in the CMIS client library are available:

- `applyAcl(List<Ace> addAcl, List<Ace> removeAcl, AclPropagation propagation)`
  First removes the ACEs (Access Control Entries) in `removeAcl` from the current ACL and adds the ACEs from `addAcl` afterward.
- `setAcl(List<Ace> acl)`
  Sets the ACL on the specified one.
- `addAcl( List<Ace> acl, AclPropagation propagation )`
  Same as applyAcl with `addAcl = acl` and `removeAcl = null`
- `remove(List<Ace> acl, AclPropagation propagation)`

Same as `applyAcl` with `addAcl = null` and `removeAcl = acl`

To modify the ACL of the current object only, set the `propagation` parameter to `OBJECTONLY`. To modify the ACL of the current object as well as of the ACLs of all of the object's descendants, set the `propagation` parameter to `PROPAGATE`. You can apply `PROPAGATE` only to folders. It works as follows: The ACEs that are added and removed at the root folder of the operation are computed and then `applyAcl` is called with these ACE sets for each descendant.

For one principal at most one ACE is stored in an object ACL. Assigning a more powerful permission to a principal replaces the inferior permission with the more powerful one. `cmis:all` is, for example, more powerful than `sap:delete`. If, for example, the current permission for a principal is `cmis:read` and the permission `cmis:write` is added this results in an ACL with one ACE for the principal containing the permission `cmis:write`. Adding an inferior permission has no effect.

Removing a permission for a principal from an object results in no ACE entry for the principal in that ACL. This is independent of the current settings in the ACL with respect to this principal.

In methods with parameters for adding and removing ACEs, first the specified ACEs are removed and then the new ones are added.

1.5.1.5.2.2 Owner Concept

Every folder and document has the `sap:owner` property. When an object is created the currently connected user automatically becomes the owner of the object. The owner of an object always has full access even without any specific ACEs granting him or her permission.

The owner property could be changed using the `updateProperties` method with the following restrictions:

- The new value of the owner property must be identical with the currently connected user.
- The currently connected user has `cmis:all` privilege.

1.5.1.5.2.3 User for Connecting to the Document Service

An application has the following options to connect to the document service:

- The application can use a connect method without explicitly providing a parameter containing a user. Then the current user is forwarded to the document service. The user's right to access particular documents and folders is determined using the user ID and the attached ACLs.
- The application can provide a user ID explicitly using a parameter of the connect method. Then this ID is used for checking the access rights.

> **i Note**
>
> Note that the document service is not connected to any Identity Provider or Identity Management System and considers the provided ID as an opaque string. This is also true for the user or principal strings provided in the ACEs when setting ACLs at objects.
>
> The application is responsible for providing the correct user ID but it can also submit a technical user ID that does not belong to any physical user, for example, to implement some kind of service user concept.

1.5.1.5.2.4 Connecting While Providing Additional Principals

Besides providing a user, some connect methods have an additional parameter to provide the IDs of additional principals to the document service.

If additional principals are provided, the user not only has his or her own permissions to access objects but in addition gets the access rights of these principles. If, for example, the user him or herself has no right to access a specific document but one of the additionally provided principals is allowed to read the content, then the user can also access the content in the context of this connection.

With this concept an application could also use roles (or even groups) in the ACLs by setting ACEs indicating these roles or groups. Then the roles of the current user can be evaluated during his connection calls and he is granted access rights according to his role (or group) membership.

It is very important to keep in mind that the additional principals are also opaque strings for the document service. This leaves it up to the application to decide what kind of information it sends as additional principals, including identifiers only known by the application itself. On the other hand, the application must ensure that there is no user with an ID similar to the additional principals, which the application uses in its ACLs because such a user might unintentionally get too many access rights.

Example

This example shows how to assign write and read permissions for two kinds of users: Authors and readers. Authors should have write access to documents and readers should only have read access to the documents. The application defines two roles, one for authors called `author-role` and one for readers called `reader-role`.

For more information about securing applications and using roles, see *Securing Applications*.

To set up permissions for authors and readers as described in our example, set the appropriate ACEs at the documents. The following code snippet shows how to set these permissions for a single document:

```
Session session = ..;
Document document = ..;

String authorRole = "author-role";
String readerRole = "reader-role";
// list of ACEs which should be added
List<Ace> addAcl = new ArrayList<Ace>();
// build and add ACE for user authors
List<String> permissionsAuthor = new ArrayList<String>();
    permissionsAuthor.add("cmis:write");
Ace aceAuthor = session.getObjectFactory().createAce(authorRole,
    permissionsAuthor);
addAcl.add(aceAuthor);
// build and add ACE for user U2
List<String> permissionsReader = new ArrayList<String>();
permissionsReader.add("cmis:read");
Ace aceReader = session.getObjectFactory().createAce(readerRole,
    permissionsReader);
addAcl.add(aceReader);
// we remove all ACEs currently set
List<Ace> removeAcl = document.getAcl().getAces();
// add and remove the ACEs at the folder
document.applyAcl(addAcl, removeAcl, AclPropagation.OBJECTONLY);
```

The next code snippet shows how the author and reader roles are automatically determined for the users and propagated to the document service to provide the users the access rights they need.

```
        import com.sap.security.um.service.UserManagementAccessor;
        import com.sap.security.um.user.User;
        import com.sap.security.um.user.UserProvider;
        …
        String authorRole = "author-role";
        String readerRole = "reader-role";

        // fetch current user and its roles
        UserProvider userProvider = UserManagementAccessor.getUserProvider();
        User currentUser = userProvider.getCurrentUser();
        Set<String> roles = currentUser.getRoles();

        // add author or reader role
        List<String> additionalPrincipals = new ArrayList<String>();

        if (roles != null && roles.contains(authorRole)){
            additionalPrincipals.add(authorRole);
        }
          else if (roles != null && roles.contains(readerRole)){
            additionalPrincipals.add(readerRole);
        }
        // connect with additional role
        String uniqueRepositoryName = ..;
        String key = ..;

        Session session = EcmFactory.connectForUser(uniqueRepositoryName,
key, null,
            currentUser.getName(), null, additionalPrincipals);
```

As long as the user's session is active, his or her permission to access the documents is determined by the user's role assignment. That is, authors can change documents and readers are only allowed to read them.

Related Information

Securing Java Applications [page 1676]

1.5.1.5.2.5 Predefined Users

There are some predefined users for the document service.

The following predefined users exist:

- The `{sap:builtin}admin` user who always has full access to all objects no matter which ACLs are set.

  > **i Note**
  >
  > Note that the document service considers user IDs only as opaque strings. Therefore, the application must prevent that a normal user connects to the document service using this administration user ID.

- The `{sap: builtin}everyone` user applies to all users. Therefore, granting a permission to this user using an ACE grants this permission to all users.

1.5.1.5.2.6 Special Rules for ACL Settings

There are some document service specific rules with respect to ACLs.

Object Creation

When creating an object the connected user becomes the owner of the new object. The ACL of the parent folder is copied to the new object and modified according to the `addAcl` and `removeAcl` parameter settings of the `create` method.

Access by Path

A user is allowed to fetch an object using the path if the user has at least the `cmis:read` permission for the object. In this case, the ACLs of the ancestor folders of the object are not relevant.

Versioning

- All documents of a version series, except the private working copy (PWC), share the same ACL and owner.
- It is only allowed to modify the ACL on the last version of a version series and only if it is not checked out.
- Principals are allowed to check out a document if they have the `cmis:write` permission for it. They become the owner of the PWC and the ACL of the PWC initially contains only one ACE with their principal name and the `cmis:all` permission.
- The ACL and the owner of a PWC can be changed independently of the other objects of the version series the PWC belongs to. Only the owner of the PWC and users with the `sap:delete` permission are allowed to check in or to cancel a checkout.
- Only principals having the `cmis:all` permission for the version series are allowed to add or remove ACEs when checking in a PWC.

Operations with Special Behavior

- `getChildren`
  Returns all children the principal is allowed to see. If the principal has no read permission for the current folder, a `NodeNotFoundException` is thrown.
- `getDecendants`
  Returns only those descendants of a folder F, which the principal is allowed to see. Only those descendants are returned for which all folders on the path from F to the descendant are accessible to the principal. If the principal has no read permission for the current folder F, a `NodeNotFoundException` is thrown.
- `getFolderTree`

Similar to `getDecendants`.

- `getFolderParent`
  If the principal has no read permission for the current folder, a `NodeNotFoundException` is thrown. If the principal has no read permission for the parent folder, a `PermissionDeniedException` is thrown.
- `getObjectParents`
  Returns a list with the parents the principal is allowed to see. Only one parent is returned because the document service does not support multi-filing. If the principal has no read permission for the current object, a `NodeNotFoundException` is thrown.
- `move`
  This method is allowed if the principal has the `sap:file` permission for the source folder, target folder, and the object to move.

1.5.1.6 Performance Tips (Java)

In many ways the document service behaves like a relational database, where each document and folder is one entry.

Therefore, most of the performance tips for databases also apply to the document service, for example:

- Use selective queries and do not fetch too many objects.
- Do not filter objects in the application if this is possible in the document service.
- Try to reduce the amount of information you request for the objects, for example, only request the ACLs or the allowed actions if you really need them. Furthermore, try to reduce the set of properties that you query to just those that your application really depends on.

To help you improve the performance of your application that uses the document service, we provide the following tips.

> **i** Note
>
> These are only recommendations, and may not be suitable in every case. There may be situations where you cannot and should not apply them.

Keep the Number of Repositories Small

Documents and folders are stored in the document service in different repositories. Creating a large number of repositories entails significant CPU usage and requires a considerable amount of storage, even if no documents are stored.

> **→** Recommendation
>
> We recommend that you keep the total number of repositories to a minimum. Avoid, for example, creating a separate repository for each user, especially if the users do not have large amounts of data to store. In such a situation, create just one repository instead and store the user data in several separate folders.

As a rule of thumb, if an application uses more than 10 repositories and if the amount of data in these repositories is small, consider using folders instead of repositories. If each repository contains a large set of data (more than 100 GB), using many repositories is not a problem.

Prevent `getChildren` Calls on Large Folders

If folders contain many children, performance might be impaired when you navigate to one of these folders using a `getChildren` call. If you navigate to a folder to analyze its data, for example, using the CMIS Workbench, this analysis becomes complicated. In contrast, fetching a child in a folder with many children by using its object ID or its path is not a problem.

It is difficult to define what qualifies as a "large" folder. If you send only one `getChildren` call per hour, then a thousand or more children would be totally acceptable, but if you send many calls per second, then even 100 children might impair performance. In any case, the load caused by calling this method increases linearly with the number of children.

Instead of having one folder with many children, you might consider subdividing the children into different subfolders or even a subfolder hierarchy. Another alternative to using the `getChildren` call option is to use the query method with the `IN_FOLDER` predicate together with additional restrictions to limit the number of matching results.

Do Not Use Large Skip Counts

Several CMIS methods have a skip count parameter, for example, the `getChildren` or the `query` method. Using large skip counts produces a significant load because a huge number of matching result objects is found and skipped before the final result set can be collected. To prevent the need for large skip counts, try to reduce the number of matching results by subdividing the children into different subfolders or by using a more selective query.

Avoid Using a Sort Criterion (for example, `getChildren`, `query`)

Only use a sort criterion if you really need it, because it might reduce performance significantly (see also *Paging with maxItems and skipCount (for example, for getChildren, query)* in the *Frequently Asked Questions*.

Do Not Request All Properties of an Object

In the operational context (see the `OperationalContext.java` class), you can define the properties that are to be returned together with the selected objects. Do not query all properties because this might be time consuming and it increases the amount of data transferred over the network. In particular, requesting the `cmis:path` property can be inefficient because it has to be computed for each call. The general rule is to

reduce the amount of data you fetch from the document service; this also applies to other information you might request together with the objects, such as the ACLs or allowed actions.

Access Using the Object ID Rather Than the Path

It is much faster to access an object using its ID than using its path.

Prevent Use of `getFolderTree`, `getDescendants`, and `IN_TREE` on Large Hierarchies

Using the `getFolderTree` or `getDescendants` method on large hierarchies is very inefficient. The same is true for the folder predicate `IN_TREE` that you can use in the statement of the query method. All these methods are slow for large hierarchies even if the final result set is small.

The reason for the performance problems with these methods is that all the descendant folders of the start folder have to be loaded from the database into the server where the document service is running. This results in many calls to the database and many objects are transferred over the network. Finally, a very complex query with all the IDs of the folders in the hierarchy has to be created and sent to the database to get the final result.

For the query method, the size of the searchable folder hierarchy is already restricted to a maximum of 1000. For larger hierarchies an exception is thrown. Be aware that even a hierarchy of 1000 folders is quite large and results in a heavy load on the system as well as bad performance for the request.

Reuse OpenCmis Session Objects

When applications use the document service they fetch a session object using one of the connect methods. Creating a session is quite an expensive operation, which should be reused and shared if possible. A session object is thread safe and allows parallel method calls.

Usually, a session is bound to a user. To reduce the number of sessions that are created, fetch the session only for the first request of the user and store it in the user's HTTP session. Then the session can be reused in subsequent requests of this user.

If an application uses a service user to connect the session to the document service, we recommend that you store this session in a central place and reuse it for all subsequent requests.

When you share a session object, observe the following tips:

- A session object has an internal cache, for example, for already fetched objects. To make sure that you fetch the latest version of specific objects, clear the cache from time to time.
- If a session is used for a very long time, problems might occur that result in exceptions (for example, network connection problems). A possible solution is to replace the failing session with a new one. However, do not replace a session if an `ObjectNotFound` exception is thrown because you tried to fetch a nonexistent document or folder. This also applies to similar situations where the exception is part of the normal method behavior.

- Do not use a single session object for a large number of requests because due to session stickiness all these requests are sent to the same server. Rather use a pool of about 50 to 100 session objects to distribute the requests over different servers on which the document service is running.

Multitenancy

One document service session is always bound to one tenant and to one user. If you create the session only once, then store it statically, and finally reuse it for all subsequent requests, you end up in the tenant where you first created the document service session. That is: You do not use multitenancy.

We recommend that you create one document service session per tenant and cache these sessions for future reuse. Make sure that you do not mix up the tenants on your side.

If you expect a high load for a specific tenant, we recommend that you create a pool of sessions for that tenant. A session is always bound to a particular server of the document service and this will not scale. If you use a session pool, the different sessions are bound to different document service servers and you will get a much better performance and scaling.

Search Hints

You can indicate hints for queries. The general syntax is:

```
hint:<hintname>[,<hintname>]*:<cmis query>
```

The following hints are currently available:

- `ignoreOwner`: Usually, documents are returned for which the current user is the owner OR is present in an ACE. The `ignoreOwner` setting returns only documents for which the current user has an ACE; ownership is ignored in this case. This improves the speed of the query because the owner check is omitted. This is useful if the owner is present in an ACE anyway.
- `noPath`: Does not return the path property even if it is requested. This improves the speed of queries on folders, because paths do not have to be computed internally.

Sample Code

```
hint:ignoreOwner,noPath:SELECT * FROM cmis:folder
```

```
hint:ignoreOwner:SELECT * FROM cmis:document
```

Related Information

Apache Chemistry OperationContext Class
Frequently Asked Questions (Java) [page 598]

1.5.1.7 Frequently Asked Questions (Java)

Answers to frequently asked questions on the document service.

How often does a backup occur?

The document service executes several backups a day to prevent file loss due to disasters. Backups are kept for 14 days and then deleted. Backups are not needed for simple hard disk crashes, since all storage hardware is based on redundant hard disks.

How to do Paging with `maxItems` and `skipCount` (for example, for `getChildren`, query)

If you implement paging using `maxItems` and `skipCount`, be aware that the different calls might be send to different database servers each returning the result objects in a possibly different order. To get a consistent result for these calls, add a unique sort criterion so that each server returns the objects using the same order. Be aware that using a sort criterion might reduce the processing speed significantly. Therefore, only use a sort criterion if really needed.

1.5.2 Consume Document Service (HTML5 Applications)

You can connect to the document service by treating it as an external service and the document service treats your HTML5 application as an external app that requests access.

Procedure

To enable external access to your document service repositories, deploy a small proxy application that is available out-of-the-box. For more information about its usage and deployment, see Access the Document Service from an HTML5 Application [page 567].

Related Information

Consuming the Document Service (Java) [page 550]

1.5.3 Managing Repositories in the Cockpit

In the cockpit, you can create, edit, and delete a document service repository for your subaccounts. In addition, you can monitor the number and size of the tenant repositories of your document service repository.

> **i** Note
>
> Due to the tenant isolation in SAP BTP, the document service cockpit cannot access or view repositories you create in SAP Document Center or the other way round.

Related Information

Create Repositories (Cockpit) [page 599]
Edit Repositories (Cockpit) [page 600]
Delete Repositories (Cockpit) [page 601]
View Content and Metadata Size of Tenant Repositories (Cockpit) [page 601]

1.5.3.1 Create Repositories (Cockpit)

In the cockpit, you can create document service repositories for your subaccounts.

Procedure

1. Log on with a user (who is a subaccount member) to the SAP BTP cockpit.
2. Choose *Repositories* > *Document Repositories* in the navigation area.
3. To create a new repository, choose *New Repository*, and enter the following data.

Field	Entry
Name	Mandatory. Enter a unique name consisting of digits, letters, or special characters. The name is restricted to 100 characters.
Display Name	Optional. Enter a display name that is shown instead of the name in the repository list of the subaccount. The name is restricted to 200 characters. You cannot change this name later on.
Description	Optional. Enter a descriptive text for the repository. The name is restricted to 500 characters. You cannot change the description later on.

Field	Entry
Virus Scan	Select this checkbox to activate the virus scan. When you create a repository, you can activate a virus scanner for write accesses. The virus scanner scans files during uploads. If it finds a virus, write access is denied and an error message is displayed. Note that the time for uploading a file is prolonged by the time needed to scan the file for viruses.
Repository Key	Enter a repository key consisting of at least 10 characters but without special characters. This key is used to access the repository metadata. You cannot recover this key. Therefore, you must be sure to remember it. You can, however, create a new key using the console client command reset-ecm-key [page 1525].
Confirm Repository Key	Reenter the repository key.

4. Choose *Save*.

Related Information

Alternative Ways to Create Repositories

Create Repositories Programmatically (Java) [page 552]

create-ecm-repository [page 1388]

Connecting Your Repository to an Application

Create Sample Applications (Java) [page 557]

1.5.3.2 Edit Repositories (Cockpit)

In the cockpit, you can change the name, key, or virus scan settings of the repository. You cannot change the display name or the description.

Procedure

1. Log on with a user (who is a subaccount member) to the SAP BTP cockpit.
2. In *Repositories* > *Document Repositories* in the navigation area, select the repository for which you want to change the name or the virus scan setting.
3. Choose *Edit*, and change the repository name or the virus scan setting.
4. Enter the repository key.
5. To change the repository key itself, choose the *Change Repository Key* button and fill in the key fields that appear.

6. Choose *Save*.

1.5.3.3 Delete Repositories (Cockpit)

In the cockpit, you can delete a repository including the data of any tenants in the repository.

Context

> ⚠ Caution
>
> Be very careful when using this command. Deleting a repository permanently deletes all data. This data cannot be recovered.

If you simply forgot the repository key, you can request a new repository key and avoid deleting the repository. For more information, see reset-ecm-key [page 1525].

Procedure

1. Log on with a user (who is a subaccount member) to the SAP BTP cockpit.
2. In *Repositories* > *Document Repositories* in the navigation area, select the repository, which you want to delete.
3. Choose *Delete*.
4. On the dialog that appears, enter the repository key.
5. Choose *Delete*.

1.5.3.4 View Content and Metadata Size of Tenant Repositories (Cockpit)

In the cockpit, you can monitor the number and size of the tenant repositories of your document service repository.

Context

If an application runs in several different tenant contexts, a tenant repository is created for each tenant context. The tenant repository is created automatically when the application connects to the document service and the respective tenant repository did not exist before.

The information on the size of a repository is updated every few hours.

Procedure

1. Log on with a user (who is a subaccount member) to the SAP BTP cockpit.
2. In *Repositories* > *Document Repositories* in the navigation area, click the name of your repository.
3. Choose *Tenant Repositories* in the navigation area.

Related Information

Tenant Context API [page 902]

1.5.4 Managing a Repository with Console Client Commands

You can create and manage repositories for the document service with client commands.

The following set of console client commands for managing repositories is available:

Related Information

Console Client Commands
add-ecm-tenant
create-ecm-repository
delete-ecm-repository
display-ecm-repository
edit-ecm-repository
list-ecm-repositories
reset-ecm-key

1.5.5 Terminate an Account and Export Data

Procedure

1. Build a proxy bridge and deploy it.

   Make sure that you set up the permissions correctly. For more information about building a proxy bridge, see Build a Proxy Bridge [page 562].
2. Download the Chemistry Workbench from the Apache Web site and connect to your proxy bridge.
3. Download the content of your repository to your local computer.

Results

To set up automated batch operations, you can use "Console" in the CMIS workbench. You can create scripts that perform queries to filter your content or you can download selective folders only. As starting point, have a look at the sample scripts that are available in the *Console* menu.

With the proxy bridge you get a standard CMIS endpoint. So you are not restricted to the CMIS workbench as only tool for export, you can use any CMIS-compliant tool.

1.6 SAP Feedback service (Beta)

The SAP Feedback service provides developers, customers, and partners with the option to collect end-user feedback for their applications. It also provides predefined analytics on the collected feedback data. This includes rating distribution and detailed text analysis of user sentiment (positive, negative, or neutral).

> **i** Note
>
> The SAP Feedback service is currently a beta offering that is available only on the SAP BTP trial landscape for trial accounts.

To use the SAP Feedback service, you must enable it from the SAP BTP cockpit for your subaccount.

The SAP Feedback service include the following components:

- Client API - collects feedback data.
- *Administration* - the user interface (UI) for administering feedback collection and feedback quota.
- *Analysis* - the UI for analyzing and exporting collected feedback data, which leverages the SAP HANA analytics and text analysis capabilities. Feedback data is stored in the SAP HANA DB.

To use the services' UIs, the following roles must be assigned to your user:

- FeedbackAdministrator
- FeedbackAnalyst

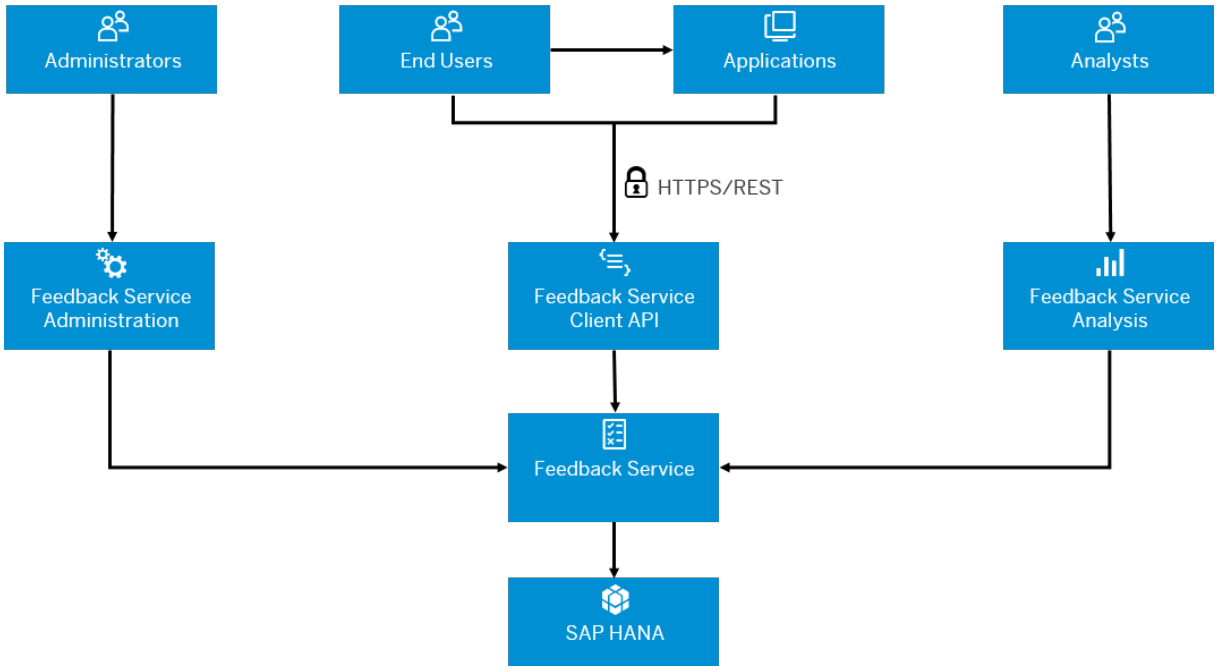
If you are a subaccount owner, these roles are automatically assigned to your user when you enable the SAP Feedback service. To enable other SAP ID users to access the *Analysis* and *Administration* UIs, you need to assign the roles manually. For more information, see Consuming the SAP Feedback service [page 605].

In the *Administration* UI, the administrator adds the applications for which feedback is to be collected. Then the developer can use the client API to consume the SAP Feedback service.

Once the SAP Feedback service is consumed by the application and feedback data is collected, the feedback analyst can explore feedback text in the *Analysis* UI. As a result, a developer can use end-user feedback to improve the performance and appearance of the specific application.

Architecture

The SAP Feedback service leverages the in-memory technology of the SAP HANA DB.



Related Information

Consuming the SAP Feedback service [page 605]
Getting Feedback for Applications [page 615]
Using Services in the Neo Environment [page 1152]

1.6.1 Consuming the SAP Feedback service

Your application can consume the SAP Feedback service either via a browser or via a Web application back end.

To enable your application to use the SAP Feedback service to collect feedback:

1. Have an SAP BTP trial account.
   You can request a free trial account on https://account.hanatrial.ondemand.com/.
2. Enable the SAP Feedback service for your trial account:
   1. In the SAP BTP cockpit navigation areas, choose *Services*.
   2. Choose *Feedback Service (BETA)* > *Enable*.
3. Assign the required roles to the users who should have access to the *Analysis* and the *Administration* UIs. By default, trial account owners are assigned to these roles. As a trial account administrator, you can assign the roles to other SAP ID users:
   1. In the SAP BTP cockpit, choose *Services* > *Feedback Service (BETA)*.
   2. Under *Service Configuration*, choose *Configure feedback* > *Roles* and assign the FeedbackAdministrator and FeedbackAnalyst roles to the appropriate users, using the following settings:
      - *Account*: *usageanalytics*
      - *Application*: *feedback*

      > **i Note**
      >
      > For the role assignments to take effect, either open a new browser session or log out from the cockpit and log on to it again.
4. In the *Administration* UI, add the application for which feedback is to be collected.
5. Modify your application code to use the SAP Feedback service client API to collect the feedback of your application users.
   Your application can consume the SAP Feedback service either via a browser or via web application back end.

Related Information

Client API [page 606]
Consume Using a Browser [page 609]
Consume Using the Web Application Back End [page 611]
Getting Feedback for Applications [page 615]
Managing Roles [page 1710]
Using Services in the Neo Environment [page 1152]

1.6.1.1 Client API

The SAP Feedback service is exposed through a client API that you can use to enable users to send feedback for your application. You do this by adding code to the application that uses the SAP Feedback service client API.

Request

An application can consume the SAP Feedback service using the service's REST API. The messages exchanged between the client (your application) and the SAP Feedback service are JSON-encoded. Call the SAP Feedback service by issuing an HTTP POST request to the unique application feedback URL that contains your application ID:

`https://feedback-`*account_name*`.hanatrial.ondemand.com/api/v2/apps/`*application_id*`/posts`

The application feedback URL is automatically generated after you register your application in the *Administration* UI of the SAP Feedback service.

Set the Content-Type HTTP header of the request to **`application/json`**. In the request body, supply a feedback resource in JSON format. The resource may have the following attributes:

SAP Feedback service Client API Attributes

Attribute	Type	Dimension	Description
texts	collection		Feedback text values
texts.tX	string	2000	The value for feedback question X. X is in the range [1 - 5]
ratings	collection		Rating values
ratings.rX	object		The rating for rating question X. X is in the range [1 - 5]
ratings.rX.value	integer	[1-5]	The value for rating question X
context	object		Feedback context
context.page	string	2038	The page for which feedback is sent
context.view	string	64	The page view for which feedback is sent
context.lang	string	2	The ISO code of the language used for the feedback text. The code is a lowercase, two-letter code as defined by ISO-639-1. The default value is `en` (English).
context.attrX	string	64	Additional context specification. X is the range [1-5].

To collect feedback data, you must provide values for at least one rating or one free-text attribute. You can additionally pass values for:

- Up to 5 rating attributes
- Up to 5 free text attributes
- Up to 8 context attributes

> ⚠ Caution
>
> According to the data privacy terms defined in the *Terms of Use for SAP HANA Cloud Developer Edition*, you must not collect, process, store, or transmit any personal data using your trial account. Therefore, do not use the context attributes of the SAP Feedback service client API to collect personal data such as user ID and user name.

Response

When the request is successful, the SAP Feedback service returns an HTTP response with code *200-OK* and an empty body.

Error Handling

In case of errors, the SAP Feedback service returns an HTTP response with an appropriate error code. Any additional information that describes the error, is contained in the response body as an *Error* object. For example:

```
{
  error: {
    code: 30,
    message: "quota exceeded"
  }
}
```

The value of *error.code* identifies the cause, and the value of *error.message* describes the cause. The string in *error.message* is not meant to be shown to your application users and is therefore not translated. The purpose of the string is to assist in the development of your application.

The table below lists the most common errors that the service can return. In addition to this list, a call to the SAP Feedback service may also result in a response with another HTTP response code. In this case, the HTTP response code itself should be enough to describe the issue.

Error Codes

Error Cause	HTTP Status Code	Content Type	error.code	error.message
Feedback quota exceeded	403	application/json	30	quota exceeded

Error Cause	HTTP Status Code	Content Type	error.code	error.message
Invalid request. All parameter values are valid, but the combination of them is not. Examples: • The request does not contain either text or a rating • A parameter is missing	400	application/json	40	`<error description>`
Invalid parameter value	400	application/json	41	invalid value for parameter `<param_name>`
Invalid JSON or empty body	400	application/json	42	`<error description>`
Incorrect service URL	404	-	-	-
Incorrect or missing Content-Type header	415	-	-	-
Incorrect HTTP method; for example, calling the SAP Feedback service with method GET	405	-	-	-
SAP Feedback service error	500	-	-	-

Example

A sample request to the SAP Feedback service may look like this:

- URL: `https://feedback-<account_name>.hanatrial.ondemand.com/api/v2/apps/<application_id>/posts`
- HTTP method: *POST*
- Content-Type: *application/json*
- Request body:

```
{
  "texts":{
    "t1": "Very helpful",
    "t2": "Well done",
    "t3": "Not usable at all",
    "t4": "I don't like it",
    "t5": "OK"
  },
  "ratings":{
    "r1": {"value":5},
    "r2": {"value":2},
```

```
    "r3": {"value":5},
    "r4": {"value":3},
    "r5": {"value":1}
  },
  "context":{
    "page": "/b2b/orders",
    "view": "payment",
    "lang": "en",
    "attr1": "1.3.15",
    "attr4": "mobile"
  }
}
```

Related Information

Consume Using a Browser [page 609]
Consume Using the Web Application Back End [page 611]
Administration [page 616]

1.6.1.2 Consume Using a Browser

Developers can consume the SAP Feedback service using a web browser.

Prerequisites

- An SAP BTP trial account.
- Install the SAP BTP Tools and create an SAP HANA Cloud server runtime environment.

Procedure

1. Create a dynamic web project:
   a. From the Eclipse main menu, navigate to *File* > *New* > *Dynamic Web Project*.
   b. In the *Project name* field, enter **`feedback-app`**. Make sure that SAP HANA Cloud is selected as the target runtime.
   c. Leave the default values for the other project settings and choose *Finish*.
2. Add an HTML file to the web project:
   a. In the *Project Explorer* view, select the *feedback-app* node.
   b. From the Eclipse main menu, navigate to *File* > *New* > *HTML File*.
   c. Enter **`index.html`** as the file name.
   d. To generate the file, choose *Finish*.

e. Replace the source code with the following content:

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta http-equiv="Content-Type" content="text/html;charset=UTF-8"/>
    <title>Feedback Application</title>
    <script src="https://sapui5.hana.ondemand.com/resources/sap-ui-core.js"
        id="sap-ui-bootstrap"
        data-sap-ui-libs="sap.m, sap.ui.commons"
        data-sap-ui-theme="sap_bluecrystal">
    </script>
    <script>
      var app = new sap.m.App({initialPage:"page1"});
      var t1 = new sap.m.Text({text: "Please share your feedback"});
      var t2 = new sap.m.Text({text: "Do you like it"});
      var ind1 = new sap.m.RatingIndicator({maxValue : 5, value : 4});
      var t3 = new sap.m.Text({text: "Some free comments:"});
      var textArea = new sap.m.TextArea({rows : 2, cols: 40});
      var sendBtn = new sap.m.Button({
          text : "Send",
          press : function() {
              var data = {
                      "texts": {t1: textArea.getValue()},
                      "ratings": {r1: {value: ind1.getValue()}},
                      "context": {page: "page1"}
              };
              $.ajax({
                  url: "https://feedback-
<subaccount_name>.hanatrial.ondemand.com/api/v2/apps/<your_application_id>/
posts",
                  type: "POST",
                  contentType: "application/json",
                  data: JSON.stringify(data)
              }).done(function() {
                  jQuery.sap.require("sap.m.MessageToast");
                  sap.m.MessageToast.show("Thank you. Your feedback was
accepted.");
              }).fail(function() {
                  jQuery.sap.require("sap.m.MessageToast");
                  sap.m.MessageToast.show("Something went wrong plese try
again later.");
              });
          }
      });
      var vbox = new sap.m.VBox({
          fitContainer: true,
          displayInline: false,
          items: [t1, t2, ind1, t3, textArea, sendBtn]
      });
      var page1 = new sap.m.Page("page1", {
          title: "Feedback Application",
          content : vbox
      });
      app.addPage(page1);
      app.placeAt("content");
    </script>
  </head>
  <body class="sapUiBody">
    <div id="content"></div>
  </body>
</html>
```

> **Note**
>
> `<Subaccount_name>` is the unique identifier that is automatically generated when the subaccount is created.

3. Adjust the service URL in the source code to point to the application feedback URL generated for your application.
4. Test the application on SAP BTP local runtime:
   a. Deploy the application on your SAP BTP local runtime.
   b. Open the application in your web browser: `http://<host>:<port>/feedback-app/`. Send sample feedback.
5. Test the application on the SAP BTP:
   a. Deploy the application on the SAP BTP.
   b. Start the application and open it in your web browser.

Related Information

Setting Up the Development Environment [page 845]
Consuming the SAP Feedback service [page 605]
Administration [page 616]
Org Administration Using the Cockpit

1.6.1.3 Consume Using the Web Application Back End

Developers can use the SAP Feedback service from the Java code in a simple Java EE Web application.

Prerequisites

- An SAP BTP trial account.
- Install the SAP BTP Tools and create an SAP BTP server runtime environment. For more information, see Setting Up the Development Environment [page 845].
- Assign the following roles to your user:
  - FeedbackAdministrator
  - FeedbackAnalyst

Procedure

1. Create a dynamic web project:

a. From the Eclipse main menu, navigate to *File* > *New* > *Dynamic Web Project*.
b. In the *Project name* field, enter **`feedback-app`**. Make sure that SAP HANA Cloud is selected as the target runtime.
c. Leave the default values for the other project settings and choose *Finish*.

2. Add a servlet to the web project:
a. In the *Project Explorer* view, select the *feedback-app* node.
b. From the Eclipse main menu, navigate to *File* > *New* > *Servlet*.
c. Enter the Java package **`hello`** and the class name **`FeedbackServlet`**.
d. To generate the servlet, choose *Finish*.
e. Replace the source code with the following content:

```
FeedbackServlet.java
package hello;
import java.io.IOException;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.entity.StringEntity;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.sap.core.connectivity.api.DestinationException;
import com.sap.core.connectivity.api.http.HttpDestination;
/**
 * Servlet implementation class FeedbackServlet
 */
public class FeedbackServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final Logger LOGGER =
LoggerFactory.getLogger(FeedbackServlet.class);
    public FeedbackServlet() {
        super();
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpClient httpClient = null;
        try {
            Context ctx = new InitialContext();
            HttpDestination destination = (HttpDestination)
ctx.lookup("java:comp/env/FeedbackService");
            httpClient = destination.createHttpClient();
            HttpPost post = new HttpPost();
            String text = request.getParameter("text");
            String rating = request.getParameter("rating");
            String page = request.getParameter("page");
            String body = "{\"texts\":{\"t1\": \"" + text + "\"}, \"ratings
\":{\"r1\": {\"value\": " + rating + "}}, \"context\": {\"page\": \"" +
page + "\", \"lang\": \"en\", \"attr1\": \"mobile\"}}";
            //Use the proper content type
            post.setEntity(new StringEntity(body, "application/json",
"UTF-8"));
            HttpResponse httpResponse = httpClient.execute(post);
            int responceCode =
httpResponse.getStatusLine().getStatusCode();
            if (responceCode != HttpServletResponse.SC_OK) {
```

```
                LOGGER.error("Feedback Service call failed with HTTP
responce code " + responceCode + ". Error: " +
httpResponse.getStatusLine().getReasonPhrase());

response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
"Something went wrong please try again later.");
            } else {
                response.getWriter().print("Your feedback was accepted.
Thank You!");
            }
        } catch (NamingException e) {
            LOGGER.error("Cannot lookup the feedback service destination",
e);

response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, "Cannot
lookup the feedback service destination");
        } catch (DestinationException e) {
            LOGGER.error("Cannot create HttpClient", e);

response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
"Something went wrong please try again later.");
        } finally {
            if (httpClient != null) {
                ClientConnectionManager connectionManager =
httpClient.getConnectionManager();
                if (connectionManager != null) {
                    connectionManager.shutdown();
                }
            }
        }
    }
}
```

3. Add an HTML file to the web project:
   a. In the *Project Explorer* view, select the *feedback-app* node.
   b. From the Eclipse main menu, navigate to *File* > *New* > *HTML File*.
   c. Enter **`index.html`** as the file name.
   d. To generate the file, choose *Finish*.
   e. Replace the source code with the following content:

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta http-equiv="Content-Type" content="text/html;charset=UTF-8"/>
    <title>Feedback Application</title>
    <script src="https://sapui5.hana.ondemand.com/resources/sap-ui-core.js"
        id="sap-ui-bootstrap"
        data-sap-ui-libs="sap.m, sap.ui.commons"
        data-sap-ui-theme="sap_bluecrystal">
    </script>
    <script>
      var app = new sap.m.App({initialPage:"page1"});
      var t1 = new sap.m.Text({text: "Please share your feedback"});
      var t2 = new sap.m.Text({text: "Do you like it"});
      var ind1 = new sap.m.RatingIndicator({maxValue : 5, value : 4});
      var t3 = new sap.m.Text({text: "Some free comments:"});
      var textArea = new sap.m.TextArea({rows : 2, cols: 40});
      var sendBtn = new sap.m.Button({
          text : "Send",
          press : function() {
              var data = {
                      "text": textArea.getValue(),
                      "rating": ind1.getValue(),
                      "page": "page1"
```

```
                };
                $.ajax({
                    url: "FeedbackServlet",
                    type: "POST",
                    data: data
                }).done(function() {
                    jQuery.sap.require("sap.m.MessageToast");
                    sap.m.MessageToast.show("Thank you. Your feedback was
accepted.");
                }).fail(function() {
                    jQuery.sap.require("sap.m.MessageToast");
                    sap.m.MessageToast.show("Something went wrong plese try
again later.");
                });
            }
        });
        var vbox = new sap.m.VBox({
            fitContainer: true,
            displayInline: false,
            items: [t1, t2, ind1, t3, textArea, sendBtn]
        });
        var page1 = new sap.m.Page("page1", {
            title: "Feedback Application",
            content : vbox
        });
        app.addPage(page1);
        app.placeAt("content");
      </script>
    </head>
    <body class="sapUiBody">
      <div id="content"></div>
    </body>
</html>
```

4. Declare a reference to the SAP Feedback service destination:
   a. Open the *WebContent/WEB-INF/web.xml* file and add the following content just before the closing web-app tag:

```
web.xml
...
<resource-ref>
    <res-ref-name>FeedbackService</res-ref-name>
    <res-type>com.sap.core.connectivity.api.http.HttpDestination</res-type>
  </resource-ref>
...
```

5. Test the application on SAP BTP local runtime:
   a. Deploy the application on the SAP BTP local runtime.
   b. Open the *Connectivity* tab page of the SAP BTP local runtime.
   c. Create a destination with the name **`FeedbackService`** and configure it to be consumed by the application at runtime.

```
Name=FeedbackService
Type=HTTP
URL=https://feedback-<subaccount_name>.hanatrial.ondemand.com/api/v2/apps/
<your_application_id>/posts
Authentication=NoAuthentication
```

   The application feedback URL, which contains the application ID, is automatically generated after you register the application in the *Administration* UI of the SAP Feedback service.
   d. Open the application in your web browser: `http://<host>:<port>/feedback-app/`. Send sample feedback.

6. Testing the application on SAP BTP:
    a. Deploy the application on the SAP BTP.
    b. Open the SAP BTP Cockpit in your web browser. Create a destination with the name **`FeedbackService`** and configure it to be consumed by the application at runtime.

```
Name=FeedbackService
Type=HTTP
URL=https://feedback-<subaccount_name>.hanatrial.ondemand.com/api/v2/apps/
<your_application_id>/posts
Authentication=NoAuthentication
```

    The application feedback URL, which contains the application ID, is automatically generated after you register your application in the *Administration* UI of the SAP Feedback service.
    c. Start the application and open it in your web browser.

Related Information

Setting Up the Development Environment [page 845]
Consuming the SAP Feedback service [page 605]
Getting Feedback for Applications [page 615]
Administration [page 616]
Configure Destinations from the Eclipse IDE [page 64]
Configure Destinations from the Cockpit [page 77]

1.6.2 Getting Feedback for Applications

After you deploy your applications on the SAP BTP, you need to add the applications for which you want to collect feedback to the *Administration* UI of the feedback service.

Adding an application generates a dedicated application feedback URL. The developer uses this URL in the client API to consume the feedback service. Once the feedback service is consumed by the application and feedback data is collected, the feedback analyst can explore ratings and text analysis in the *Analysis* UI. Developers can then use the feedback to improve the application performance and appearance.

To use the *Administration* and *Analysis* UIs, you must be assigned the following roles:

- FeedbackAdministrator
- FeedbackAnalyst

As a subaccount owner, the roles are automatically assigned to your user after you enable the feedback service. To allow other SAP ID users to access the *Analysis* and *Administration* UIs, you need to assign the roles manually.

You can also provide your feedback about the feedback service and its UI. Choose the *Feedback* button and share your ideas and suggestions for improvement. Information about your landscape host as well as about the specific place (page, view, or tab) from which you have called the feedback form is collected by SAP for analysis purposes.

Related Information

Administration [page 616]
Feedback Analysis [page 617]
Managing Roles [page 1710]
Consuming the SAP Feedback service [page 605]

1.6.2.1 Administration

An administrator can configure the feedback service.

As a feedback administrator, you can:

- Add applications for which feedback is to be collected in the *Administration* UI of the feedback service
- Customize descriptions of feedback questions
- Customize descriptions of context attributes
- Free up feedback quota space

Once you add an application to your list, you enable it to use the feedback service. As a result, a URL that is specific to both the subaccount and the application is generated. To start collecting feedback, the developer integrates the URL into the application UI, to enable end users to post feedback (for example, in a feedback form). The URL is called through a POST request by the application that wants to send feedback. That is, once an end user submits the feedback form, the application calls the feedback service using the URL and the service stores the user feedback.

An application feedback URL looks similar to the following:

```
https://feedback-<subaccount_name>.hanatrial.ondemand.com/api/v2/apps/
<application_id>/posts
```

To use the *Administration* UI of the feedback service, you need to be assigned the FeedbackAdministrator role.

To access the *Administration* UI, open the following URL in your browser:

```
https://feedback-<subaccount_name>.hanatrial.ondemand.com/admin/mobile
```

Each subaccount has a feedback quota assigned, that is, a specific amount of feedback data that can be stored in the SAP HANA DB. The quota is 250 feedback forms filled in by end users. When you reach 70% of the feedback quota, you see a warning message. Once you reach the limit, the feedback service stops processing feedback requests and storing feedback data, until you free up quota space. Do this by deleting the feedback records for a specific time period.

For each application, you can edit the description of:

- Rating questions
- Free text questions
- Context attributes

Descriptions appear in the *Analysis* UI.

If you have the FeedbackAnalyst role assigned (in addition to the FeedbackAdministrator role), you can analyze feedback results and export raw feedback data.

Related Information

Administer Application Feedback [page 617]
Feedback Analysis [page 617]
Managing Roles [page 1710]

1.6.2.1.1 Administer Application Feedback

As a feedback administrator, you can add applications and administer application feedback.

Procedure

1. Open the *Administration* UI, where you can perform the following tasks:
   a. Add an application by choosing the *+Add* button and enter a name for the application for which feedback is to be collected.
   b. To customize the description of a rating, a free text question, or a context attribute, click the pencil icon in the respective attribute row.
   c. To free up quota space, click the *Free Up Quota Space* link and choose a specific time period for which to permanently delete feedback data.
2. Save your changes.

1.6.2.2 Feedback Analysis

As a feedback analyst, in the *Analysis* UI of the SAP Feedback service you can explore the feedback collected from end users by viewing detailed ratings or text analysis, or exporting the feedback text as raw data.

The rating analysis presents information about rating questions and how feedback rating is distributed according to time and distribution criteria.

Select Time Period

You can choose a specific time period for which to view analyzed feedback data and to export raw data. The default time period is the last 7 days.

Export Raw Data

You can export raw feedback data, so that you can perform specific analysis tailored to your needs. You download raw feedback data in a .CSV format encoded in UTF-8.

> **i** Note
>
> If there are characters that do not appear correctly when you open the exported file, reopen it as UTF-8 encoded.

Related Information

Free-Text Questions [page 618]
Rating Questions [page 619]
SAP HANA Developer Guide
Client API [page 606]
Administration [page 616]

1.6.2.2.1 Free-Text Questions

As a feedback analyst, you can explore the feedback collected from end users by viewing the detailed text analysis. Text analysis classifies user feedback by:

- Type (request, problem)
- Sentiment (positive, negative, or neutral)

For further information about text analysis, read the *Text Analysis* section in the *SAP HANA Developer Guide*.

Get an Overview of Feedback Text Sentiment

The *Overview* screen displays a summary of all free text feedback questions. Each question tile provides the following information:

- Question text description
- Sentiment summary in %

The sentiment summary provides a useful overview of negative, positive, and neutral sentiments of user feedback. Feedback from a single user can result in a small or large amount of the overall sentiment count of the specific question. In other words, sentiment is calculated not per user feedback but by the sentiment elements (words) in the feedback text.

View Question Details

Select a question tile to see detailed information about the question, including the following:

- Full question text
- Number of feedback responses
- Filter that enables you to narrow down the responses to a specific text analysis group, that is, to feedback of specific type or sentiment
- A list of all feedback responses with short details about feedback type, sentiment, and date posted

For example, you can filter your responses for a specific question to show only feedback of type *Problem* that has *Negative* and *Neutral* sentiment. The returned list is ordered by date (most recent is on top).

> **i** Note
>
> No matter what filter is applied, the list always includes responses (if any) that are not classified by type or sentiment.

View Feedback Details

You can drill down to see details about a specific feedback response and examine the actual feedback text analysis. You can view the entire response with all detected text analysis "hits". In addition, you can choose the types of "hits" to highlight within the text. For example, you can choose to highlight just the *Problem* type that has *Negative* and *Neutral* text analysis. Alternatively, you can remove all highlights.

Related Information

SAP HANA Developer Guide
Client API [page 606]
Administration [page 616]

1.6.2.2.2 Rating Questions

As a feedback analyst, you can examine the feedback collected from users by viewing a detailed rating analysis.

Users can reply to each rating question by choosing a number on the scale of 1 to 5 where 1 is the lowest rating and 5 is the highest.

Get an Overview of Feedback Rating Questions

The Overview screen shows a summary of all rating questions. Each question tile provides the following information:

- Question text description
- Average rating

View Question Details

Select a question tile to see detailed information about the question during the time period you specified, including the following:

- Full question text
- Average rating received
- Number of feedback responses
- Two graph views of feedback rating distribution
- Two table views of feedback rating distribution

Depending on the time period, the graph and table views show the following data:

- Feedback distribution by rating: A graph or table showing the percentage of the overall feedback responses that receive a specific rating number. That is, how feedback is distributed in terms of a specific rating.
- Feedback distribution by time period: A graph or table in of feedback distribution among various time frame granularities, for example, a day or a year. The data shown is the average rating for the specified time granularity and applies only to the time period initially selected.

Related Information

SAP HANA Developer Guide
Client API [page 606]
Administration [page 616]

1.6.3 Data Protection and Privacy

Governments place legal requirements on industry to protect data and privacy. We provide features and functions to help you meet these requirements.

> **i Note**
>
> SAP does not provide legal advice in any form. SAP software supports data protection compliance by providing security features and data protection-relevant functions, such as blocking and deletion of personal data. In many cases, compliance with applicable data protection and privacy laws is not covered by a product feature. Furthermore, this information should not be taken as advice or a recommendation

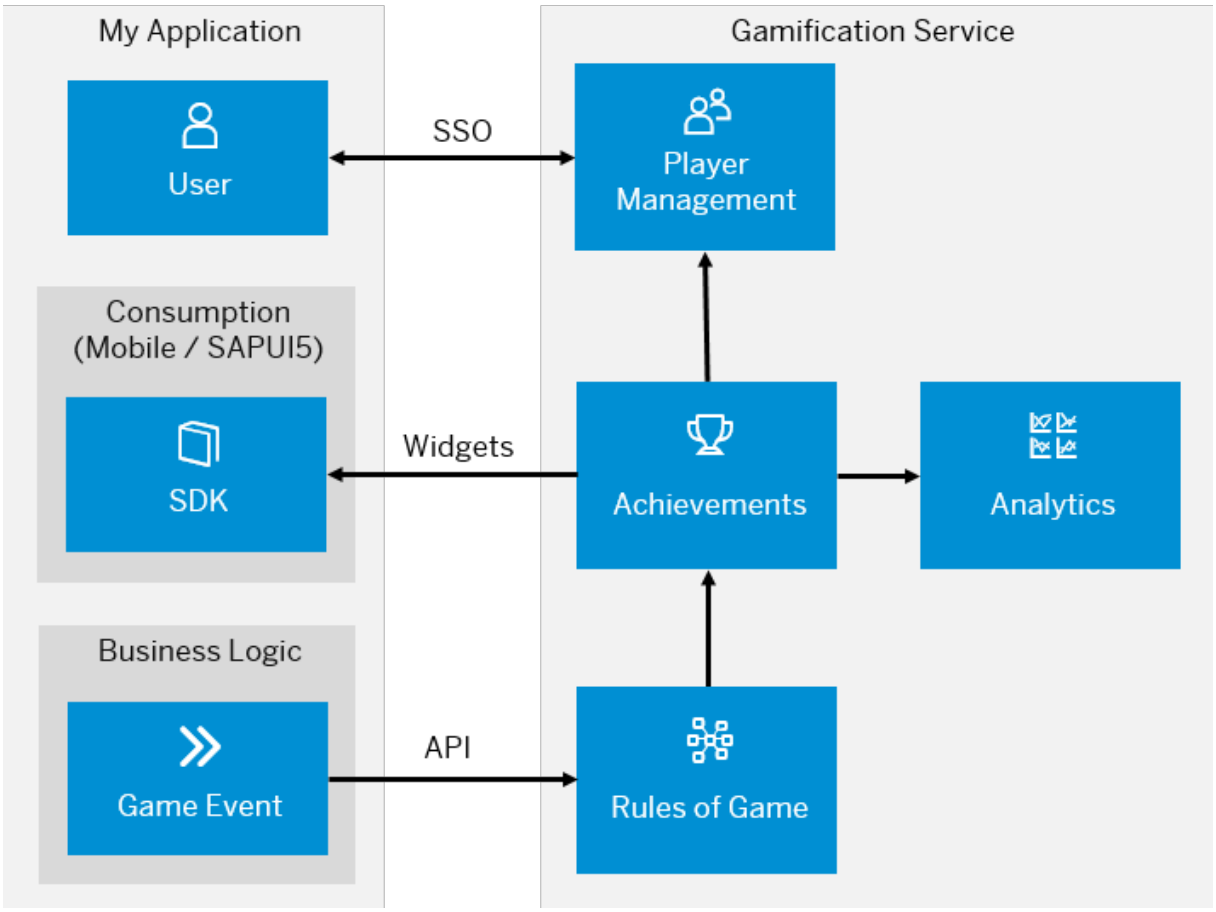
regarding additional features that would be required in specific IT environments. Decisions related to data protection must be made on a case-by-case basis, taking into consideration the given system landscape and the applicable legal requirements. Definitions and other terms used in this documentation are not taken from a specific legal source.

For more information, see Data Protection and Privacy.

1.7 Gamification

Overview

The Gamification allows the rapid introduction of gamification concepts into applications. The service includes an online development and operations environment (gamification workbench) for implementing and analyzing gamification concepts. The underlying gamification rule management provides support for sophisticated gamification concepts, covering time constraints, complex nested missions, and collaborative games. The built-in analytics module allows you to perform advanced analysis of the player's behavior to facilitate continuous improvement of game concepts.



Pain Points Addressed

- Effort for introducing gamification in new and existing apps
- Limits of achievement systems and existing platforms regarding the complexity of supported game mechanics and the speed of feedback
- The management of sophisticated gamification concepts, meeting enterprise performance, security, and scalability requirements

Product Features

- Web-based IDE (gamification workbench) for modeling game mechanics and rules
- Gamification engine for real-time processing of sophisticated gamification concepts involving time constraints and cooperation
- Built-in runtime game analytics for continuous improvement of game designs
- Web API for integration
- Simple SAPUI5 integration based on widgets
- Single-sign-on (SSO) support based on Identity Authentication
- Enterprise-level performance and scalability

Related Information

Getting Started [page 622]
Gamification Development Cycle [page 632]
Gamification Workbench [page 634]
Security [page 638]
Managing Apps [page 641]
Configuring Game Mechanics [page 646]
Integrating Gamification into a Target Application [page 681]
Analyzing Gamification Concepts [page 689]
Case Study: Gamified Help Desk Application [page 691]

1.7.1 Getting Started

Learn how to enable the gamification service in your subaccount, and how to configure and use the sample application *HelpDesk*.

1. Enable Gamification [page 623]
2. Assign Gamification Roles [page 624] (Automated)
3. Configure Destinations [page 625] (Automated, credential of technical user must be provided)

4. Enable Principal Propagation [page 627] (Automated)
5. Generate Demo Content for HelpDesk [page 628]
6. Use the Gamified HelpDesk Application [page 628]

When enabling the service, configuration steps 2, 3, and 4 are executed automatically, as follows:

- All gamification roles are assigned to the user who enabled the service.
- The required destinations are created at the subaccount level. The destination `gsdest` requires credentials (user/password). In the trial version you can use an SCN, it is safer to create a dedicated technical user.

> **i Note**
>
> If you use your SCN user to configure `gsdest`, make sure you change the destination configuration after you've changed the SCN user password in SAP ID Service. Otherwise, your user will be locked when using the HelpDesk app.

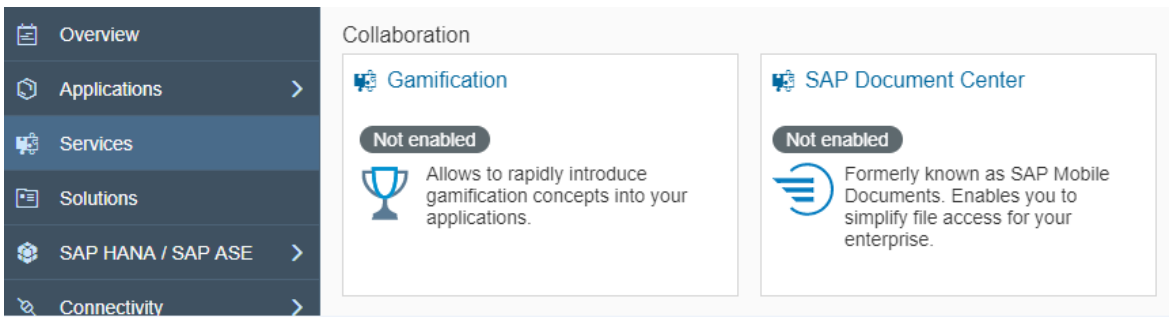
1.7.1.1 Enable Gamification

Prerequisites

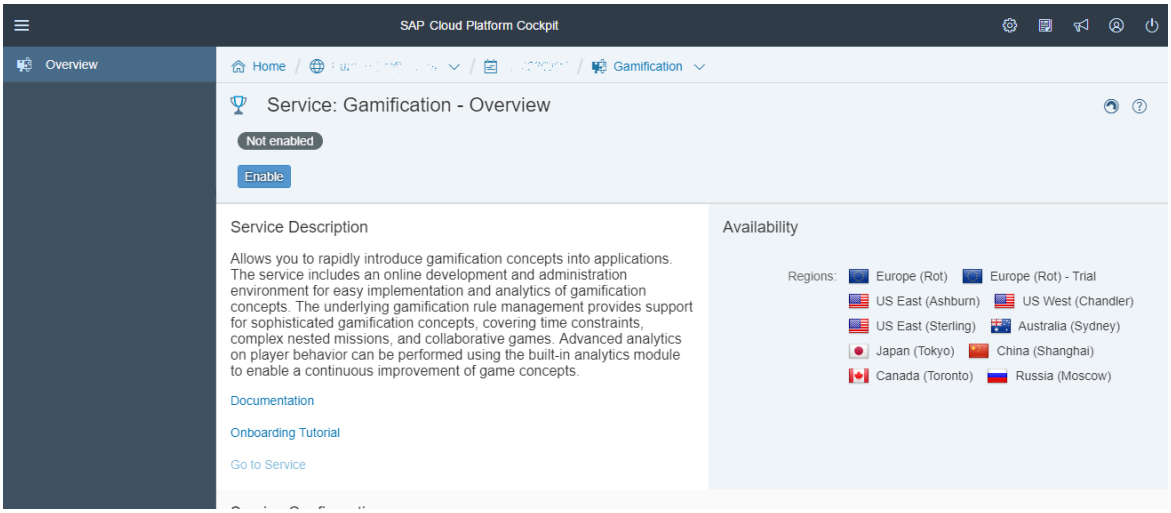
- Access to a SAP BTP account for personal development, or to a Trial account.
- A subaccount for which you are assigned the role `Administrator`.
- An SCN user.

Procedure

1. In the SAP BTP cockpit, choose the *Services* tab.
2. Select *Gamification Service*.



3. From the service's detail view choose *Enable*.



1.7.1.2 Assign Gamification Roles

Prerequisites

Log in to the SAP BTP cockpit using your SCN user and password.

Procedure

1. In the SAP BTP cockpit, choose the *Services* tab.
2. Choose the *Gamification Service* tile.
3. Choose the *Configure Gamification Service* link.
4. Go to the *Roles* tab.
5. Assign the necessary roles, for example `AppStandard`, `AppAdmin`, `GamificationDesigner`, `TenantOperator`, and `helpdesk`.

Related Information

Managing Roles [page 1710]

1.7.1.3 Configure Destinations

Prerequisites

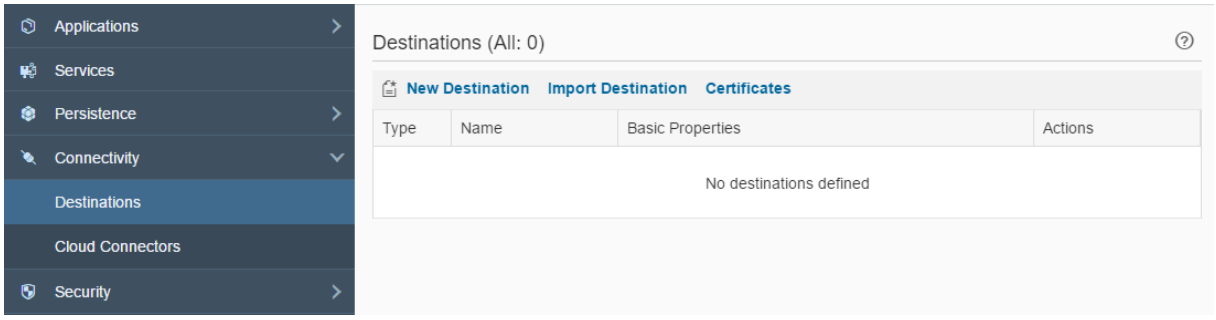
Log in to the SAP BTP cockpit using your SCN user and password.

Context

You must configure a destination to allow the communication between your application (in this case, a sample app) and your subscription to the gamification service. For the sample application, two destinations are necessary:

- **`gsdest`** – enables the gamification service API to send the events
- **`gswidgetdest`** – enables the gamification service API to request the achievements

> **i Note**
>
> Create these destinations at the subaccount level of your personal user account.



1.7.1.3.1 Create a Destination for Sending Events

Procedure

1. In the cockpit, choose the *Destinations* subtab from the *Connectivity* tab.
2. Enter the name: **`gsdest`**.
3. Select the type: **`HTTP`**.
4. (Optional) Enter a description.
5. Enter the application URL of your service instance: **`https://<application_URL>/gamification/api/tech/JsonRPC`**

   You can find the application URL of your service instance by navigating to *Subaccount* > *Services* > *Gamification Service* > *Go to Service*.

6. Select the proxy type: **`Internet`**.
7. Select **`Basic Authentication`**.
8. Enter a user ID. *Recommendation*: Use a separate technical user, see following procedure. Alternatively, you can use your SCN user. In this case make sure to update the destination as well in case of password changes. Otherwise the SAP ID Service locks your user when you attempt to use the HelpDesk app.
9. Enter the SCN password.
10. Choose *Save*.

Related Information

HTTP Destinations [page 91]

1.7.1.3.2 Create a Destination for Getting Players' Achievements

Procedure

1. In the cockpit, choose the *Destinations* subtab in the *Connectivity* tab.
2. Enter the name: **`gswidgetdest`**.
3. Select type: **`HTTP`**.
4. (Optional) Enter a description.
5. Enter the application URL of your service instance: **`https://<application_URL>/gamification/api/user/JsonRPC`**

   You can find the application URL of your service instance by navigating to *Subaccount* > *Services* > *Gamification Service* > *Go to Service*.
6. Select proxy type: **`Internet`**.
7. Select authentication: **`AppToAppSSO`**.
8. Choose *Save*.

Related Information

HTTP Destinations [page 91]

1.7.1.4 Enable Principal Propagation

Prerequisites

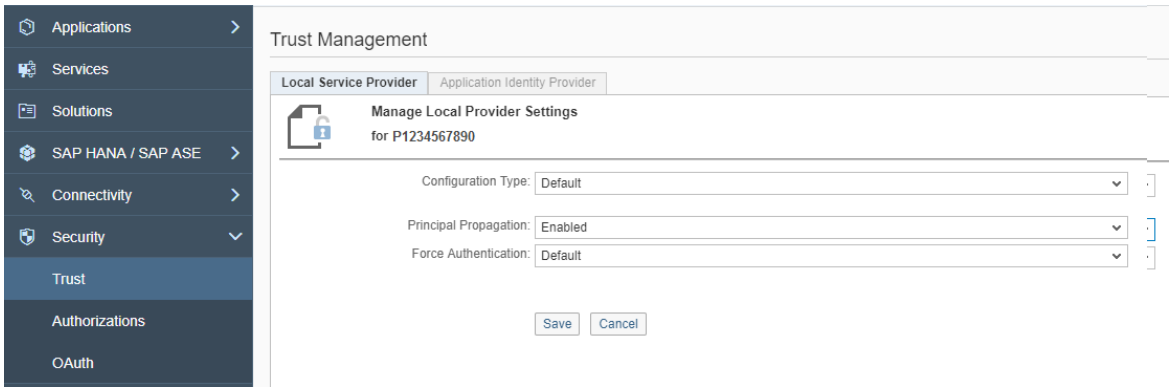
- Log in to the SAP BTP cockpit using your SCN user and password.
- A subaccount for which you are assigned the role `Administrator`.

Context

To support application-to-application SSO as part of destination **`gswidgetdest`**, you must configure your subaccount to allow principal propagation.

Procedure

1. Open the cockpit and choose the *Trust* subtab from the *Security* tab.
2. Choose the *Local Service Provider* subtab.
3. Choose *Edit*.
4. Change the *Principal Propagation* value to **`Enabled`**.



Related Information

Application-to-Application SSO Authentication [page 104]

1.7.1.5 Generate Demo Content for HelpDesk

Prerequisites

- Log in to the SAP BTP cockpit with your SCN user and password.
- You are assigned the role `TenantOperator`.

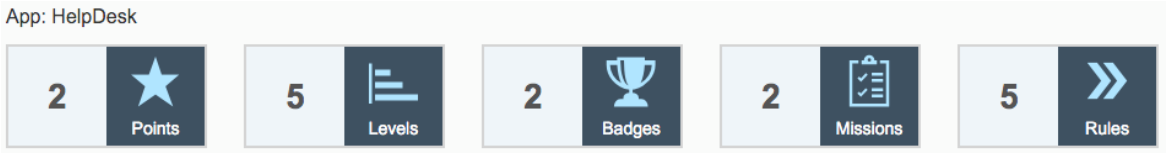
Procedure

1. In the SAP BTP cockpit, choose the *Services* tab.
2. Go to *Gamification Service* and select the *Go to Service* icon.
3. In the gamification workbench select *Operations*.
4. Go to *Data Management* and select *Create HelpDesk Demo*.



Eventually you see a notification: “Gamification concept successfully created.”

5. Switch to the *HelpDesk* application by using the dropdown box in the upper-right corner.
6. On the *Summary* tab, verify that all game mechanics are available.



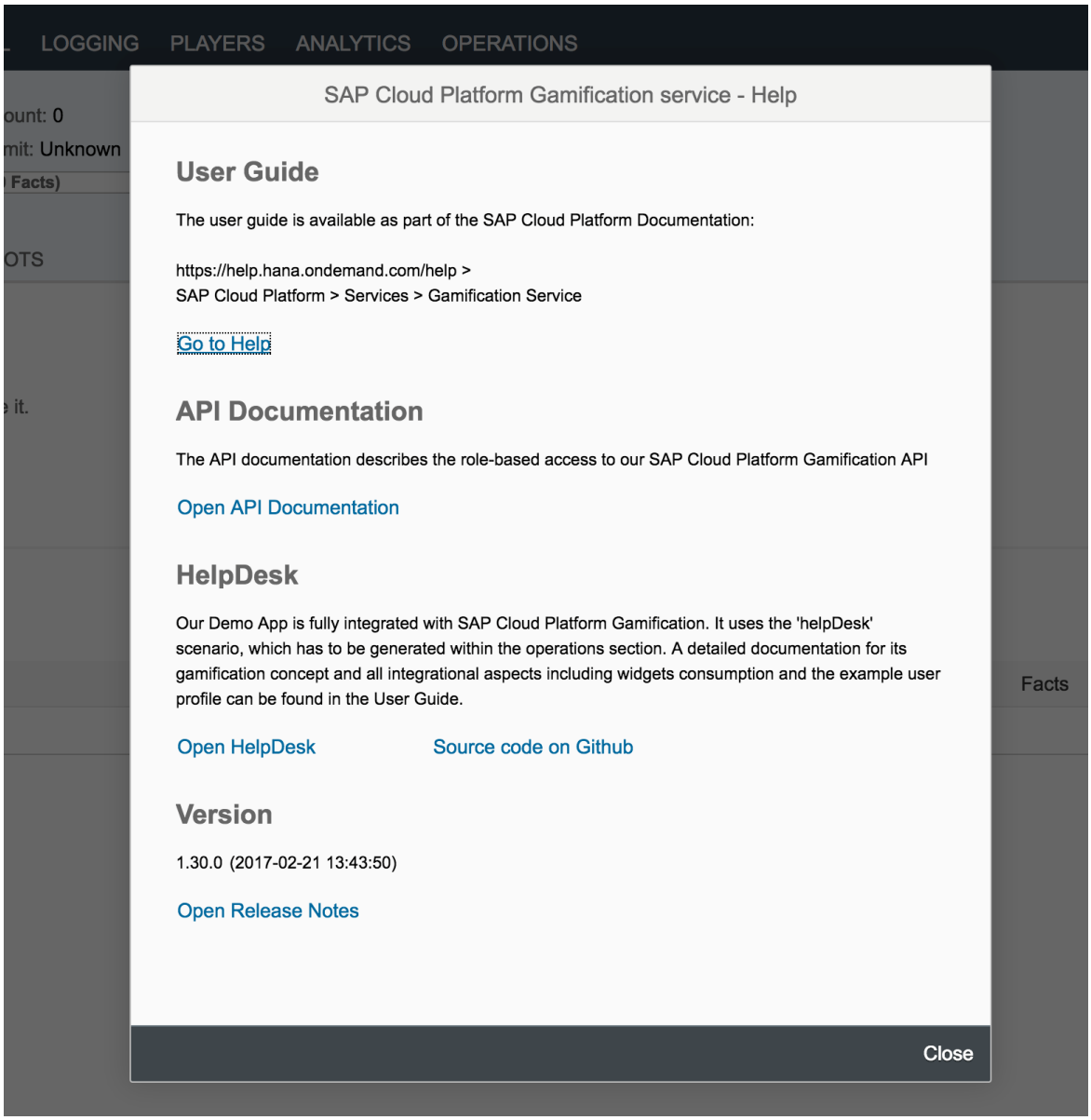
1.7.1.6 Use the Gamified HelpDesk Application

Prerequisites

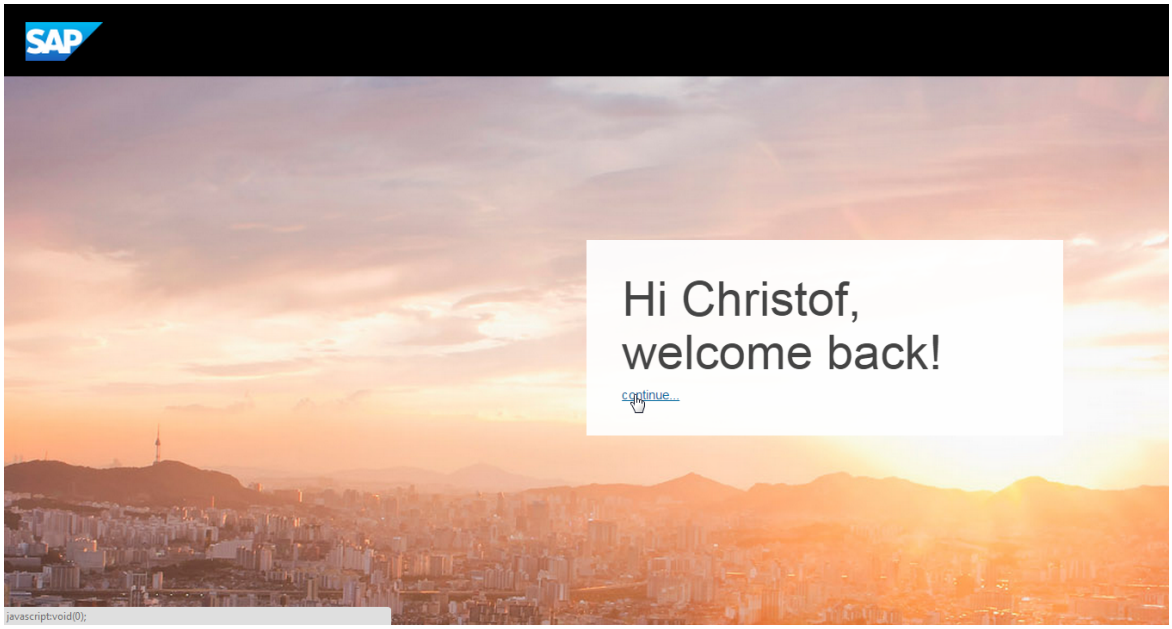
- You are assigned the role `helpdesk`.
- **HelpDesk** demo content.
- The destinations **`gsdest`** and **`gswidgetdest`** are available.

Procedure

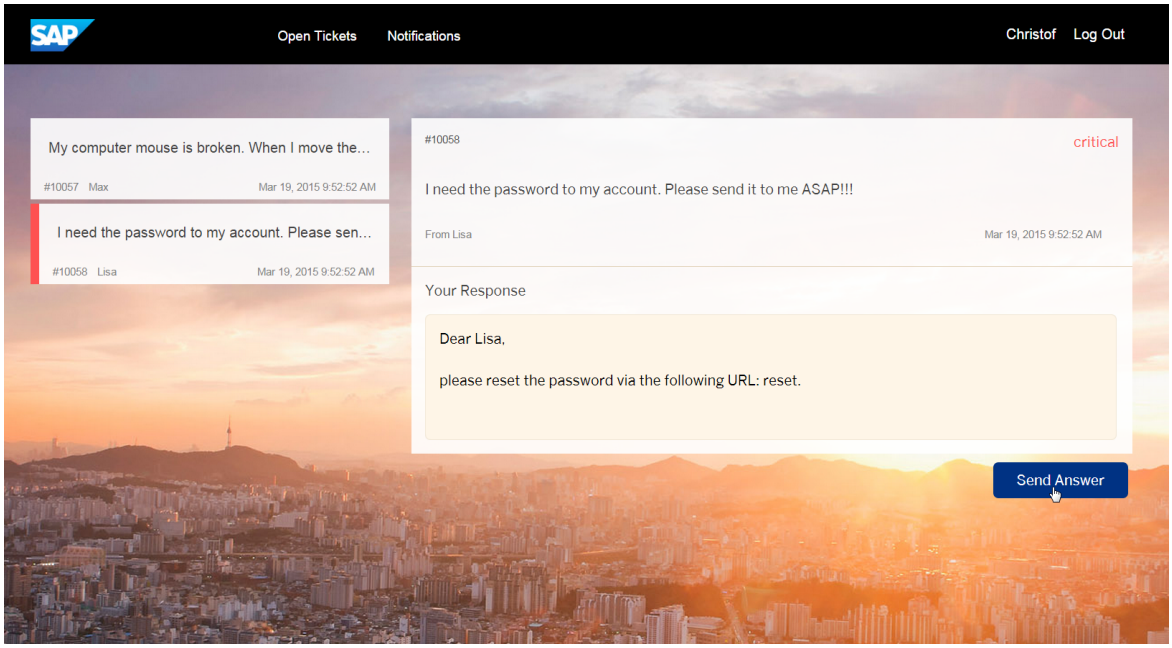
1. In the SAP BTP cockpit, choose the *Services* tab.
2. Go to *Gamification Service* and choose the *Go to Service* link.
3. Select the *Help* link in the upper right corner.
4. Select *Open HelpDesk*.



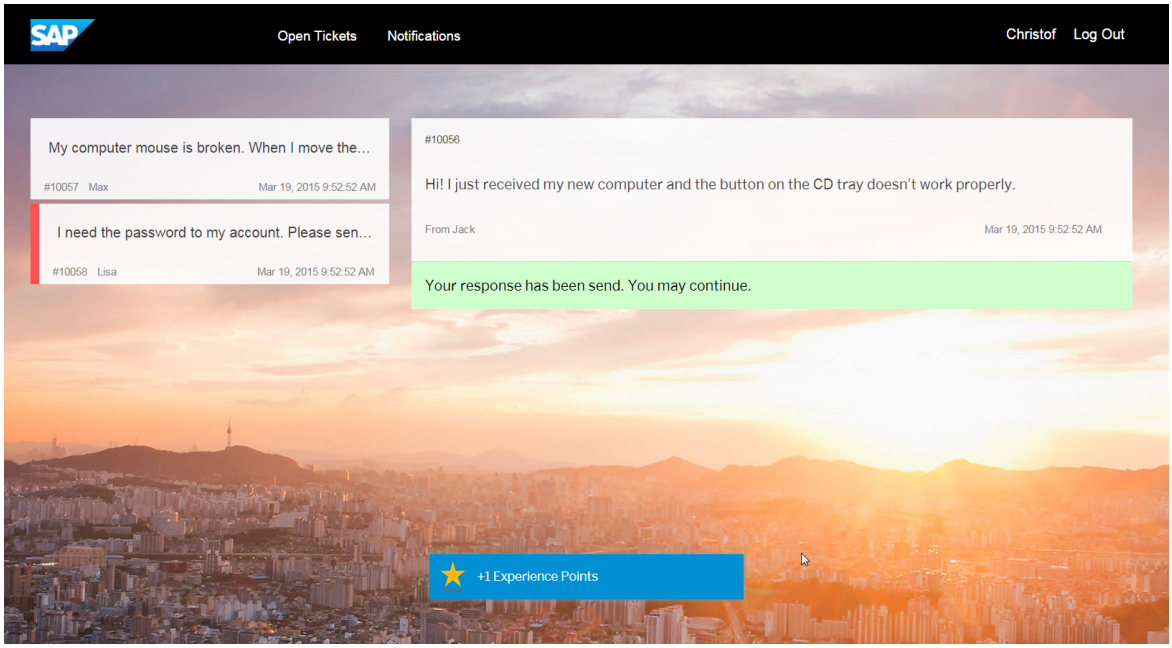
5. Log in the **HelpDesk** application.



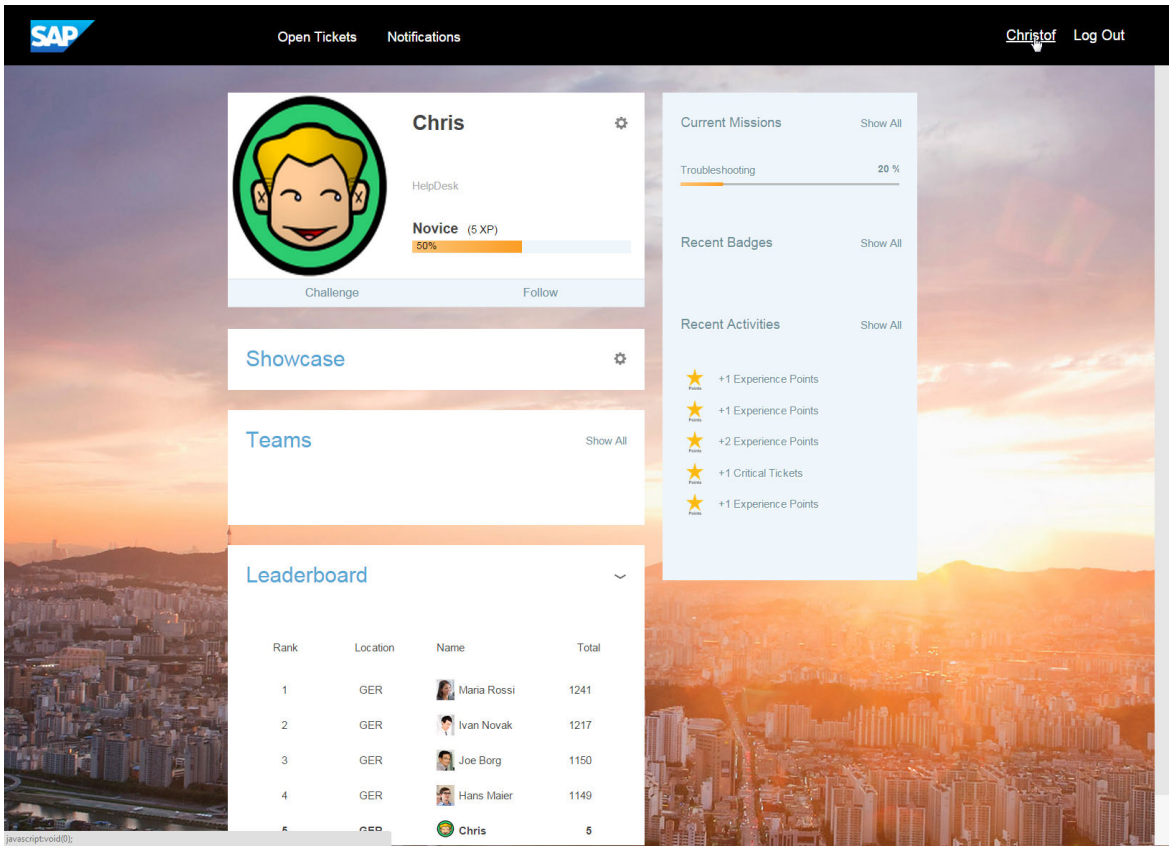
6. Process a ticket.



7. You receive points when you complete various activities.



8. Review your user profile.



1.7.2 Gamification Development Cycle

The gamification development cycle describes to introduce gamification into existing or new applications.

At a high level, the following processes are required:

- Create the gamification concept
- Implement the concept
- Integrate the concept into the target application

Creating gamification concepts is purely a conceptual tasks that is typically executed by gamification designers. The task is executed during the design phase and covers the specification of a meaningful game / gamification design.

Implementing the concept mans mapping it to the mechanics offered by the gamification service. This task is also normally performed by gamification designers, or IT experts.

Integration is a development task that includes the technical integration of the target application with the APIs of the gamification service. This is normally performed by application developers, since it requires technical knowledge of the application (such as implementing points for listening for events or creating visual representation of achievements).

1.7.2.1 Creating a Gamification Concept

A gamification concept, normally developed by gamification designers and domain experts describes the mechanics that will encourage users (players) to perform certain tasks. For example an award system comprising point and badges to encourage call center employees to process tickets efficiently or to select more complex tickets over more straightforward ones.

> **i** Note
>
> Creating gamification concepts is not a service that is covered or supported by the gamification service.

A simple gamification concept includes elements such as points and badges. For example, users are awarded experience points for certain actions, and badges as a visual representation. The gamification concept describes how these elements motivate users. It therefore includes descriptions of the actions (within the application) that allow users to attain the various achievements.

Additional examples include missions that foster collaboration or activities with time constraints that encourage users to work faster.

Related Information

Gamification Design [page 696]

1.7.2.2 Implementing the Gamification Concept

Implementation means mapping a gamification concept to the elements used in the gamification service. You can use the gamification workbench to maintain the gamification elements, such as points, badges, levels, or rules. You can modify the gamification concept at runtime.

Gamification is about full transparency to users, and is intended to encourage them. We therefore advise against modifying a concept significantly without informing users, since doing so might catch them by surprise and could possibly demotivate them.

Related Information

Configuring Game Mechanics [page 646]

1.7.2.3 Integration with the Application

Integration refers to the technical integration of the target application with the APIs of the gamification service. Integration is required to send events that are of interest to the gamification service, for example, when a user in a call center has successfully processed a ticket. Integration is also necessary to notify the users about their achievements, to send notifications to users for earned points, or to display user profiles.

The gamification service supports the integration of mainly cloud applications running with SAP BTP. Integration of other applications is technically possible, but restricted for security reasons.

Related Information

Integrating Gamification into a Target Application [page 681]

1.7.2.4 Analyzing Gamification Concepts

Gamification is a continuous process. It is crucial that you monitor the influence of a gamification concept and react to the users' behavior. For example, you want to know if your gamification concept motivates the target group or if users lose interest.

The gamification service offers basic analytics: for example, the assignment of points or badges to users over time. Therefore, you can analyze peaks and troughs of user achievements.

Related Information

Analyzing Gamification Concepts [page 689]

1.7.2.5 Ensuring Data Privacy

The introduction of gamification often requires the acquisition of sensitive information. For example you might need to track user behavior within an application to allow the gamification of onboarding scenarios.

The gamification service lets you anonymize user data. It also offers secure communication via the various APIs. However it is ultimately the responsibility of the host application to ensure data privacy however, and application developers must ensure that only the necessary data is sent to the gamification service.

Related Information

Integrating Gamification into a Target Application [page 681]

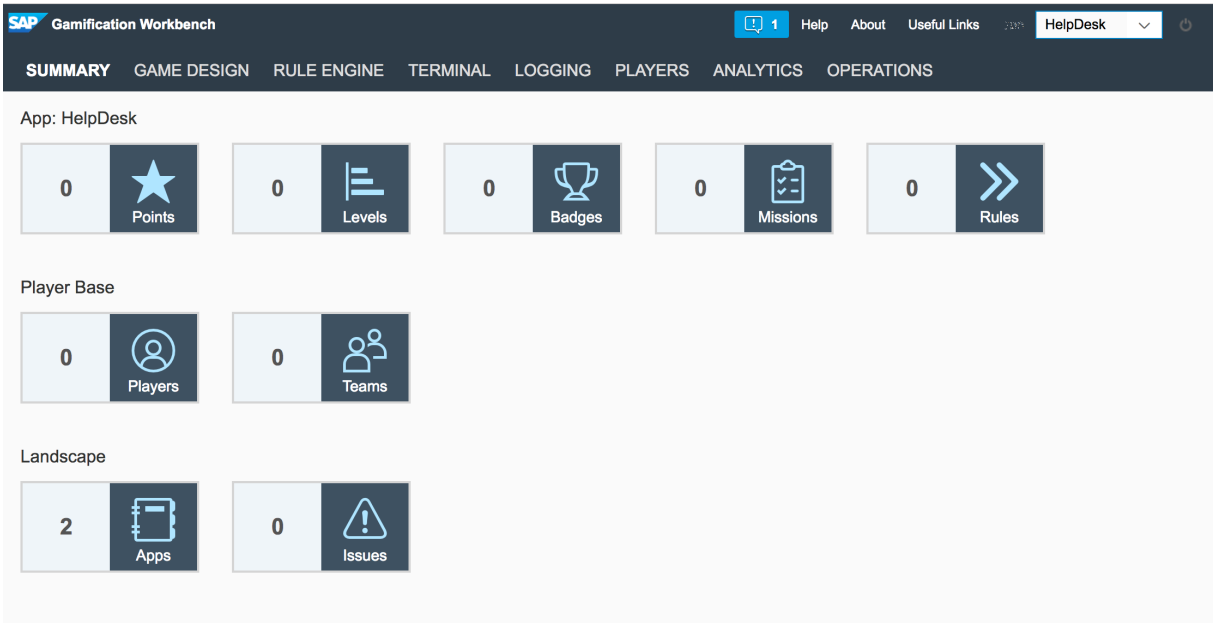
1.7.3 Gamification Workbench

The gamification workbench is the central point for managing all gamification content associated with your subaccount and for accessing key information about your gamification usage. It allows you to manage the

gamification concepts of all applications deployed in your subaccount from a single dedicated Web-based user interface.

Summary Dashboard

The figure below shows an example of the *Summary* dashboard in the workbench and is followed by an explanation:



The entry page *Summary* of the gamification workbench provides an overview of the gamification concept for the selected app, the overall player base and overall landscape.

Logon

You can log on with your subaccount user via SSO (single-sign on).

The gamification workbench can be accessed using the *Subscription* tab in the SAP BTP cockpit. The following link will be used: https://< SUBSCRIPTION_URL>/gamification .

Navigation

The navigation menu comprises the following elements:

- Summary
- Game Design
- Rule Engine

- Terminal
- Logging
- Players
- Analytics
- Operations

> **i Note**
>
> You must have specific roles in order to access the gamification workbench, see Roles [page 636].

The following table describes the navigation levels in more detail:

Level	Description
Summary	• Displays a game design summary of the currently selected app
Game Design	• Allows you to read and configure game mechanics (managing points, badges, levels, missions and rules for example) for multiple applications
Rule Engine	• Allows `Gamification Designers` and `Tenant Operators` to monitor the rule engine of the currently selected app
Terminal	• Allows you to test the gamification concept using the API
Logging	• Allows `Gamification Designers` and `Tenant Operators` to track player and rule activities
Players	• Allows you to manage gamification service users • Allows you to view gamification user profiles
Analytics	• Allows you to view gamification statistics (achievements gained by players for example)
Operations	• Allows you to manage apps • Allows you to create demo content • Allows you to import and export the gamification concept

1.7.3.1 Roles

Different roles can be assigned to users, to enable them to explicitly access the gamification workbench.

Related Information

Security [page 638]

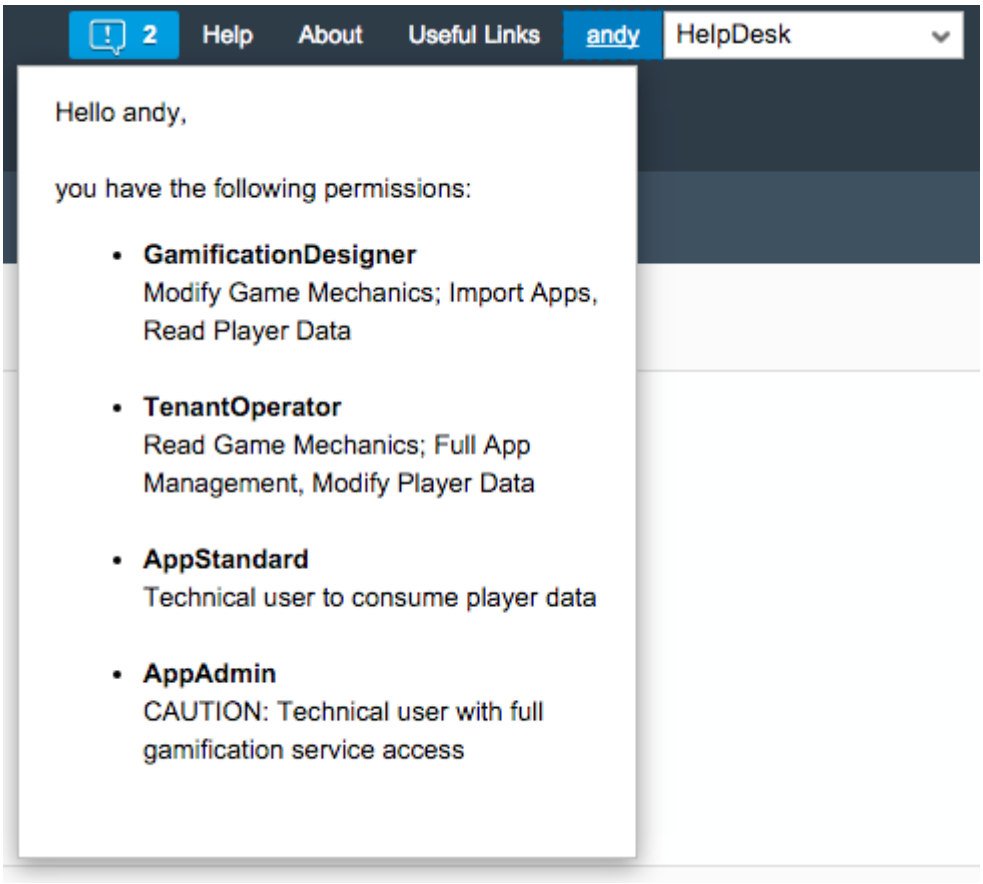
1.7.3.2 View Assigned User Roles

Prerequisites

- You have logged on to the gamification workbench.
- At least one gamification service role is assigned to your user.

Procedure

1. Hover the cursor over your user name.
2. Wait until the user details are displayed.



1.7.4 Security

Context

The gamification service offers the gamification workbench, an API for integration and a demo app. The access to the user interfaces and API is protected using SAP BTP roles.

> **i Note**
>
> Roles must be explicitly assigned to a SAP BTP user.

> **i Note**
>
> The API can be used for the integration of host applications. For productive use a technical user (SAP BTP user) should be created for a communication between the host application and the gamification service. (The use of a personal subaccount or user is only recommended for testing or demo purposes.)

1.7.4.1 Roles

The following roles can be assigned to access the gamification service gamification workbench, API or demo app and must be explicitly assigned to a SAP BTP user:

Role	Type	Access Level	Description
GamificationDesigner	User	Workbench • Summary • Game Design • Rule Engine • Terminal • Analytics	• Full access to game mechanics and rules (read, write, activate etc.) for all apps created for the tenant • Partial access to the Rule Engine monitor, to read information about the health state of the current app's rule engine • Access to the AP Terminal, allowing to test the gamification concept directly via API • Read aggregated gamification analytics (no access to individual player data)

Role	Type	Access Level	Description
GamificationReviewer	User	Workbench • Summary • Game Design (read-only) • Analytics	• Read game mechanics and rules (no write access) • Read aggregated gamification analytics (no access to individual player data) • Export game mechanics and rules for a certain app
TenantOperator	User	Workbench • Game Design (read-only) • Rule Engine • Logging • Players • Operations	• Managing Apps • Full access to the Rule Engine monitor with the ability to restore session snapshots • Read game mechanics and rules (no write access) • Access to the AP Terminal, allowing to test the gamification concept directly via API • Full access to user data, including player details with player achievements • Full access to team data • Full tenant export and import, including player data

Role	Type	Access Level	Description
AppStandard	Technical	API (methods are annotated with required role) Terminal (send events for testing purposes)	• Write only - using rules; reading achievements is possible, but should be avoided • Send player-related events • Read player achievements and available achievements Sample use cases: • Show aggregated gamification statistics in a host application • Visualize specific leaderboards in a host application
AppAdmin	Technical	API (methods are annotated with required role)	• Read and delete a player record for a single app or for the whole tenant • Create and delete a user or a team
Player (automatically assigned)	Technical (implicit role)	API (methods are annotated with required role)	• Send player-related events (only works for the user that is authenticated using the identity provider which is configured for your subaccount) **i Note** This role is not a standard SAP BTP role. It is automatically assigned to a user (player) that is created using the gamification service and cannot be explicitly assigned to a SAP BTP user.
helpdesk	User	Demo App	• Access to demo app HelpDesk

1.7.4.1.1 Assign Roles

Prerequisites

You have logged on to the SAP BTP cockpit.

Procedure

1. In the SAP BTP cockpit, choose the *Services* tab .
2. Go to *Gamification Service*.
3. In the *Service Configuration* panel choose *Configure Gamification Service*.
4. Choose the *Roles* section.
5. Assign role (`GamificationDesigner` or `TenantOperator`).

Related Information

Managing Roles [page 1710]

1.7.4.2 Data Privacy

The Gamification meets the security and data privacy standards of the SAP BTP. In general, the gamification service is not responsible for any content such as game mechanics or player achievements. It is the responsibility of the host application to meet any local data privacy standards. Therefore, you need to make sure that the personal information of players is protected according to the local regulations. In some cases where the gamification is applied to employee scenarios work council approval for the gamified host application might be necessary.

1.7.5 Managing Apps

Prerequisites

You have the role `TenantOperator`, are logged into the gamification workbench, and have opened the *Apps* tab in the *Operations* section.

Context

The gamification service introduces the concept of apps. An app represents a self-contained, isolated context for defining and executing game mechanics such as points, levels, and rules.

All data or meta data associated with an app are stored in an isolated way. In addition to this, an isolated rule engine instance is created and started for each app.

> **i** Note
>
> Players are stored independently from apps and can therefore take part in multiple apps.

1.7.5.1 Configuring Apps

Prerequisites

You have the roles `TenantOperator` and `GamificationDesigner`, are logged into the gamification workbench, and have opened the *Apps* tab in the *Operations* section.

Context

An app represents a self-contained, isolated context for defining and executing game mechanics.

Create Apps

Procedure

1. Press *Add* to add a new app.
2. Enter a name, which serves as unique identifier.
3. Optional: Enter a description.
4. Optional: Enter an app owner which can serve as contact for tenant operators.
5. Optional: Set *Auto-Create Players* flag: if set to true, then players are created automatically on first event.
6. Press *Create* to add a new app. The app should now appear in the app selection combo box in the upper right corner of the gamification workbench.

Update Apps

Procedure

1. Select the app in the list to be updated.
2. Press *Edit*.
3. Change the values of the attributes that shall be updated.
4. Press *Save*.

Delete Apps

Procedure

1. Select the point category in the list to be deleted.
2. Press *Delete*.

1.7.5.2 Switch Apps

Prerequisites

You have the role `GamificationDesigner` or `TenantOperator` or both and are logged into the gamification workbench.

Context

By switching the app, the gamification workbench only shows game mechanics and player achievements associated with the selected app.

Procedure

1. Select an app in the *app selection* combo box located in the upper right corner of the gamification workbench.
2. Optional: Review whether the app has been changed successfully, for example by comparing the summary page (tab *Summary*).

1.7.5.3 Export Apps

Prerequisites

You have the role `TenantOperator`, are logged into the gamification workbench and have opened the *Operations* tab, and navigated to the *Data Management* section.

Context

The gamification service allows exporting **all** available apps including their content. You can choose between a full tenant export including all player data and an export of game mechanics only. The latter can be imported again.

Procedure

1. Select the *Export* mode in the combo box labeled *Export* in the form area *Import / Export*.
   - Full Export: export all game mechanics and player data.
   - Game Mechanics: export game mechanics only.
2. Press *Download* to start the export. Your browser should show the file storing dialog.
3. Store the provided ZIP file on your disk.

1.7.5.4 Import Apps

Prerequisites

- You have the role `TenantOperator`, are logged into the gamification workbench and have opened the *Operations* tab, and navigated to the *Data Management* section.
- You have a gamification service export file.

  > **i Note**
  >
  > See section Exporting Apps [page 644] for details.

Context

The gamification service allows importing game mechanics based on existing gamification service export files (ZIP format). Section *Exporting Apps* explains how to do the export.

Procedure

1. Press *Browse* in the form area *Import / Export* to select the import file.
2. Press *Upload* to start the import based on the selected file.

> **i Note**
>
> If an app with the same name already exists, the import will skip this app and does not overwrite its content.

3. Apply rule changes to active rules again.

> **i Note**
>
> See section Configuring Rules [page 659] for details.

1.7.5.5 Generate Demo Apps and Content

Prerequisites

You have the role `TenantOperator`, are logged into the gamification workbench, and have opened the *Operations* tab, and navigated to the *Data Management* section.

Context

The gamification service is shipped with selected demo content comprising game mechanics as well as demo players. The demo content is created within the context of a new app.

Procedure

1. Press *Create HelpDesk Demo*.
2. An app called in the same ways as the demo content should now appear in the app selection combo box in the upper right corner of the gamification workbench.

> **i Note**
>
> Appropriate content (points, levels, badges, and rules) is created for the app automatically.

1.7.5.6 Delete Demo Apps and Content

Prerequisites

You have the role `TenantOperator`, are logged into the gamification workbench and have opened the *Operations* tab, and navigated to the *Data Management* section.

Context

The gamification service is shipped with selected demo content comprising game mechanics as well as demo player. The demo content is created within the context of a new app. The app can be deleted manually, but this will not delete generated demo players. To delete the full demo content, the explicit action must be triggered.

Procedure

1. Press the *Delete HelpDesk* button and confirm the dialog.
2. A notification should appear, stating that the demo content has been deleted.

1.7.6 Configuring Game Mechanics

Prerequisites

You have the `GamificationDesigner` role , are logged on to the gamification workbench and have opened the *Game Design* tab.

Context

The gamification concept describes the metrics, achievements and rules that are applied to an application. The following checklist describes the tasks required to implement your gamification concept in your subscription of the gamification service.

1. Configuring Achievements:
   - Configuring Points (Point Categories) [page 647]
   - Configuring Levels [page 649]

- Configuring Badges [page 651]
- Configuring Missions [page 653]

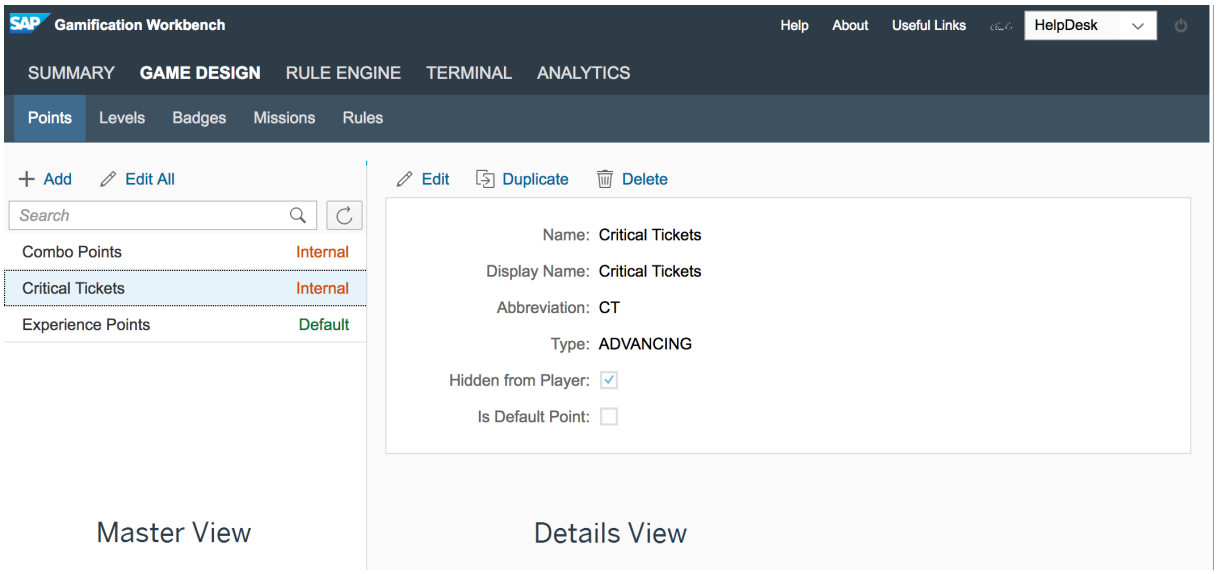
2. Configuring and Managing Rules [page 659]

General Procedure

For each game mechanics entity there is a tab with a master and details view.

- Master View
  - Shows the list of available entities.
  - *Add* button for adding a new entity.
  - *Edit All* button for switching to batch deletion mode.
- Details View
  - Shows entity attributes and images.
  - *Edit* button for editing entity attributes.
  - *Duplicate* button for cloning the complete entity including attribute values.
  - *Delete* button for deleting the given entity.

Each entity has at least the attributes name and a display name. The name serves as the unique identifier and is immutable.



1.7.6.1 Configuring Points (Point Categories)

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Points* tab.

Context

Points are the fundamental element of a gamification design. For example, points can indicate the progress in various dimensions. Points can be flagged as "Hidden from Player" for security or privacy reasons. Points that are flagged as hidden are not visible to players. Instead they can be utilized in rules. Furthermore points can have various different subtypes. The table lists the available point types.

Point Types

Type	Description
ADVANCING	Advancing points are points that can never decrease. They are used to reflect progress.
REPUTATION	Reputation points should be used within a predefined range. For examplem, from -100 to 100.
AUXILIARY	Auxiliary points can be used to define specific metrics and should only be presented to the user within their context. For example, auxiliary points can be used to track mission progress.
OTHER	Any points that do not match a specific type.

Points can be configured in the *Points* subtab of the *Game Design* tab.

1.7.6.1.1 Create Points

Prerequisites

You have the `GamificationDesigner` role, are logged on to the gamification workbench and have opened the *Points* tab.

Procedure

1. Press *Add* to add a new point category.
2. Enter a name, which serves as an unique identifier.
3. Optional: Enter a display name, which can be used to display the point to the player.
4. Enter an abbreviation which can be used to display the point to the player.
5. Select Point Type.
6. Optional: Select if the points are hidden from players.
7. Select if the point is the default point. There can only be one default point within one app.
8. Press *Create*.

1.7.6.1.2 Update Points

Prerequisites

You have the `GamificationDesigner` role, are logged on to the gamification workbench and have opened the *Points* tab

Procedure

1. Select the point category in the list to be updated.
2. Press *Edit*.
3. Change the values of the attributes that will be updated.
4. Press *Save*.

1.7.6.1.3 Delete Points

Prerequisites

You have the `GamificationDesigner` role, are logged on to the gamification workbench and have opened the *Points* tab

Procedure

1. Select the point category in the list to be deleted.
2. Press *Delete*.
3. Confirm deletion by pressing *Confirm* in the popup dialog.

1.7.6.2 Configuring Levels

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Levels* tab.

> ⚠ Caution
>
> Only levels that are based on the default point category are exposed to the default user profile.

Context

A level describes the status of a user once a specific goal is reached. The gamification service allows you to define levels based on a defined point category. The threshold defines the value of the selected point type to reach the level.

1.7.6.2.1 Create Levels

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Levels* tab.

Procedure

1. Press *Add* to add a new level.
2. Enter a name, which serves as an unique identifier.
3. Optional: Enter a display name, which can be used to display the level to the player.
4. Select the point category on which the level is based on. (The primary player levels are typically based on the default point.)
5. Enter a point threshold. Reaching it will complete the level for the player.
6. Optional: Choose an inmage for the level.
7. Press *Create*.

1.7.6.2.2 Update Levels

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Levels* tab.

Context

Procedure

1. Select the level in the list to be updated.
2. Press *Edit*.
3. Change values of attributes that shall be updated.
4. If a level has an image check *Delete Image* checkbox to remove the image.
5. Press *Save*.

1.7.6.2.3 Delete Levels

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Levels* tab.

Procedure

1. Select the level in the list to be deleted.
2. Press *Delete*.
3. Confirm deletion by pressing *Confirm* in the popup dialog.

1.7.6.3 Configuring Badges

Prerequisites

You have logged onto the gamification workbench with the role `GamificationDesigner` and you have opened the *Badges* tab.

Context

A badge is a graphical representation of an achievement. Hidden badges are not visible to the user before the assignment and can be used as surprise achievements.

1.7.6.3.1 Create Badges

Prerequisites

You have logged onto the gamification workbench with the role `GamificationDesigner` and you have opened the *Badges* tab.

Procedure

1. Press *Add* to add a new badge.
2. Enter a name, which serves as an unique identifier.
3. Optional: Enter a display name, which can be used to display the badge to the player.
4. Optional: Enter a description, which will hold information how to recieve this badge.
5. Choose an image for the badge.
6. Select if the badge is hidden from players.
7. Press *Create*.

1.7.6.3.2 Update Badges

Prerequisites

You have logged onto the gamification workbench with the role `GamificationDesigner` and you have opened the *Badges* tab.

Procedure

1. Select the badge in the list to be updated.
2. Press *Edit*.
3. Change values of attributes that shall be updated.
4. Press *Save*.

1.7.6.3.3 Delete Badges

Prerequisites

You have logged onto the gamification workbench with the role `GamificationDesigner` and you have opened the *Badges* tab.

Procedure

1. Select the badge in the list to be deleted.
2. Press *Delete*.
3. Confirm deletion by pressing *Confirm* in the popup dialog.

1.7.6.4 Configuring Missions

Prerequisites

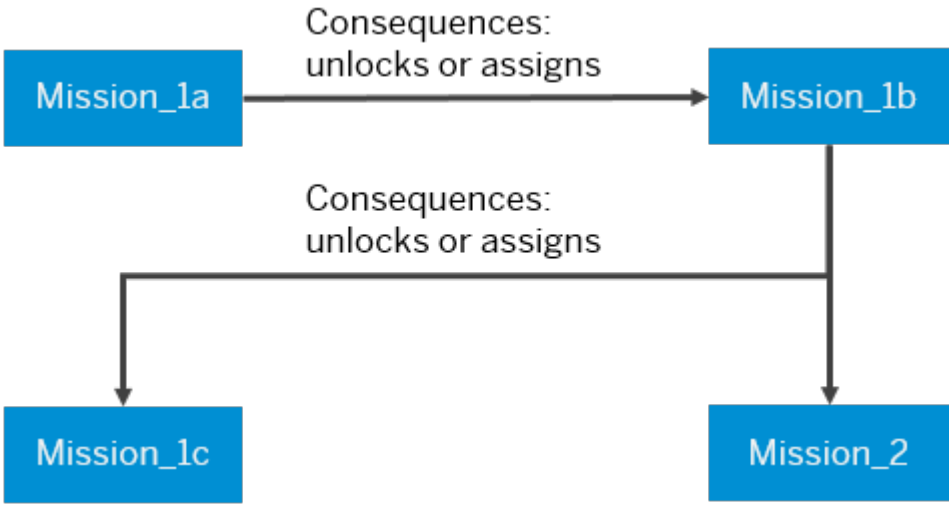
You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Missions* tab.

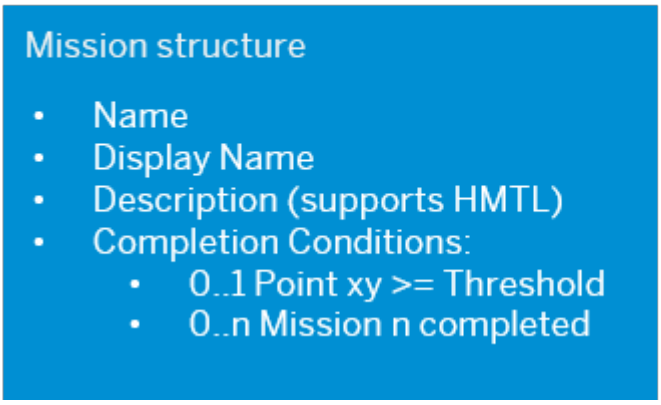
Context

A mission defines what has to be achieved to gain a measurable outcome. Besides basic standalone missions the gamification service allows modelling complex mission structures using mission conditions and consequences.

> **i** Note
>
> Mission conditions and consequences are of descriptive nature only. Actual condition checking and the execution of consequences has to be done by corresponding rules. These rules are not generated automatically yet.

Sample structure of complex missions:





Currently two types of completion conditions are supported:

- Point Conditions: A number of points, each with a respective threshold. Each point can be considered as a progress indicator: As soon as the threshold is reached, the condition is met.
- A list of missions that have to be completed. Within the API such missions are referred to as sub missions.

The consequences part is limited to a list of follow-up missions, which should be assigned or unlocked after the current mission has been completed. Within the API such follow-up missions are referred to as `nextMissions`.

Example for a rule that checks a point condition in its WHEN part and assigns a follow-up mission in its THEN part:

- WHEN

```
$p : Player($playerid : id)
eval(queryAPIv1.hasPlayerMission($playerid, 'Troubleshooting', false) == true)
eval(queryAPIv1.getScoreForPlayer($playerid, 'Critical Tickets', null,
null).getAmount() >= 5)
```

- THEN

```
updateAPIv1.completeMissionForPlayer($playerid, 'Troubleshooting', null);
updateAPIv1.assignMissionToPlayer($playerid, 'Troubleshooting reloaded',
null);
```

1.7.6.4.1 Create Missions

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Missions* tab.

Procedure

1. Press *Add* to add a new mission.
2. Enter a name that will serve as a unique identifier.
3. Optional: Enter a display name, which can be used to display the mission to the player.
4. Enter a meaningful description, what is the mission about.
5. Optional: Define different point conditions for the mission. For each condition a specific point and threshold must be specified.
6. Optional: Add required missions by pressing the *Add* button and selecting them in the pop-up window. Required missions have to be completed as precondition for completing the given mission. In the API these missions are referred to as “sub missions”.
7. Optional: Define follow-up missions by adding missions to the Assigns Missions section. In the API these missions are referred to as “next missions”.
8. Press *Create*.

Results

> **i Note**
>
> Adding a sub mission or follow-up mission only creates relations in the database. The corresponding rules for checking conditions, assigning follow up missions, or both are not generated yet. They have to be created manually. But without storing these relationships and making them available through the achievement query API it would not be possible to create such rules at all.

1.7.6.4.2 Update Missions

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Missions* tab.

Procedure

1. Select the mission in the list to be updated.
2. Press *Edit*.
3. Change the values of the attributes that shall be updated.
4. Press *Save*.

1.7.6.4.3 Delete Missions

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Missions* tab.

Procedure

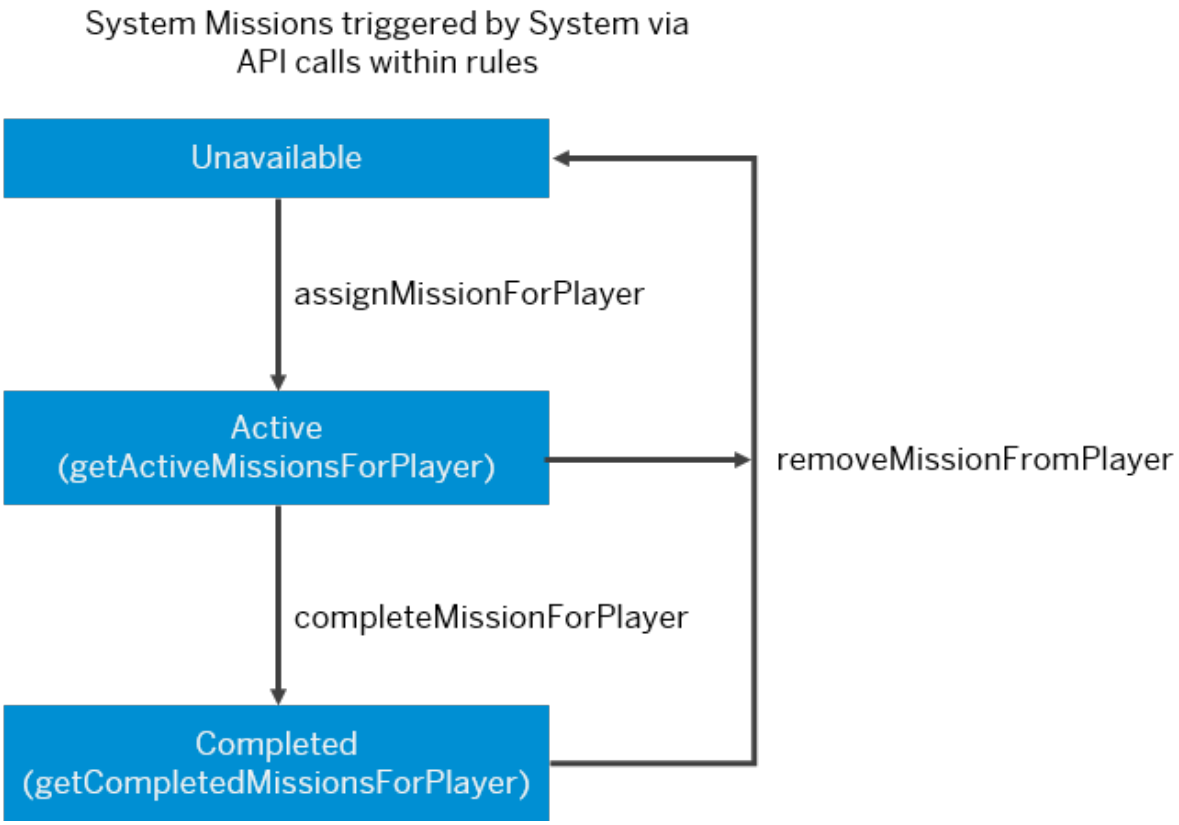
1. Select the mission in the list to be deleted.
2. Press *Delete*.
3. Confirm deletion by pressing *Confirm* in the popup dialog.

1.7.6.4.4 Controlling the Mission Life Cycle

Regarding the mission life cycle,there are two types of missions:

- System Missions: the mission life cycle is fully controlled by the service using API calls within rules.
- User-accepted Missions: the player actively decides whether to accept or reject missions, while the remaining mission life cycle (unlocking or completing a mission) is controlled by the service. In both cases the API calls have to be executed within rules to ensure data consistency between the engine and the backend.

System Missions



All state transitions are triggered by calling the respective API methods within rules, while the list of missions in a certain state can be retrieved either by calling the API directly or within a rule.

Sample rule for assigning a system mission as part of the *user init* rule:

- WHEN

```
$event : EventObject(type=='initPlayerForApp', $playerid : playerid) from
entry-point eventstream
```

- THEN

```
updateAPIv1.assignMissionToPlayer($playerid, 'Troubleshooting', null);
```

Sample rule for completing a system mission:

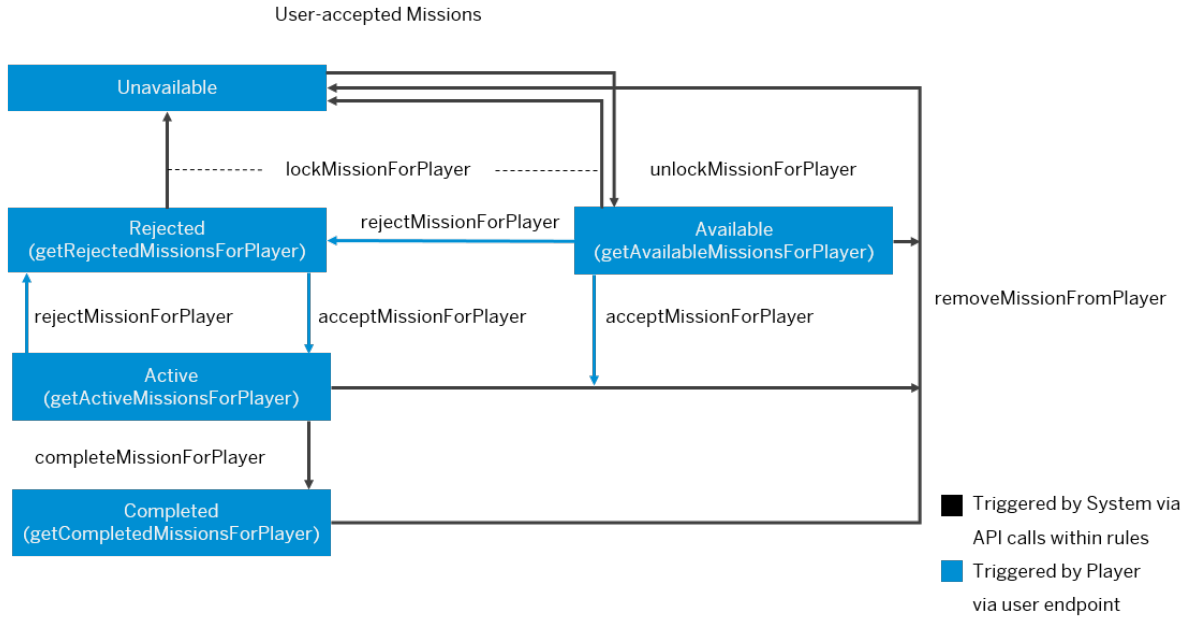
- WHEN

```
$p : Player($playerid : id)
eval(queryAPIv1.hasPlayerMission($playerid, 'Troubleshooting', false) == true)
eval(queryAPIv1.getScoreForPlayer($playerid, 'Critical Tickets', null,
null).getAmount() >= 5)
```

- THEN

```
updateAPIv1.completeMissionForPlayer($playerid, 'Troubleshooting', null);
```

User-accepted Missions



Flow for accepted missions

1. System via rule: `unlockMissionForPlayer` (allowing users to start missions).
2. User via API (User Endpoint): `getAvailableMissionsForPlayer` (listing or offering missions for the user to pick from).
3. User via (User Endpoint): `acceptMissionForPlayer` (accepting a specific mission).
4. (User progresses while having this set as active mission).
5. System via rule: `completeMissionForPlayer` (when all conditions are met, the rule completes the mission).

Flow for rejected missions

1. System via rule: `unlockMissionForPlayer` (allowing users to start missions).
2. User via API (User Endpoint): `getAvailableMissionsForPlayer` (offering specific missions for the user to pick from).
3. User via API (User Endpoint): `rejectMissionForPlayer` (rejecting an available mission).
4. Mission is no longer available for the user and will not be offered again.
5. (Optional) User via API (User Endpoint): acceptMissionForPlayer so that rejected missions become active again.

> **i Note**
>
> Invoking the manual mission methods via the user endpoint currently does not trigger any rules. If there is a rule that has to trigger when missions become active for players it would require a separate event to trigger this rule.

1.7.6.5 Configuring and Managing Rules

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Rules* tab.

Context

The rules are a fundamental element of the game mechanics. They describe the consequences of actions, the corresponding constraints and the goals that can be achieved. The rules allow you to define complex conditions and consequences based on common complex event processing (CEP) operators.

Related Information

Rules Language [page 660]
http://docs.jboss.org/drools/release/5.6.0.Final/drools-expert-docs/html/ch05.html

1.7.6.5.1 Rules Design Principles

Rules are the core elements of the gamification design. Generally they follow the event condition action (ECA) structure as for active rules in event driven architectures. Each rule is structured in two parts:

- Left hand side (LHS): rule conditions or trigger (events conditions and/or player conditions)
- Right hand side (RHS): rule consequences (updates from the player and/or event generation)

The rule conditions (LHS) are maintained in the *Trigger ("when")* area. Examples are:

- Incoming event type - wait for event of type "solvedProblem":

```
$event : EventObject(type=='solvedProblem', $playerid:playerid) from entry-
point eventstream
```

- Conditions on event parameters - player has more than 10 "Experience Points":

```
$event : EventObject(type=='solvedProblem', data['relevance']=='critical',
$playerid:playerid) from entry-point eventstream
```

- Conditions on player achievements - player has more than 10 "Experience Points":

```
eval(queryAPI.getPointsForPlayer($playerid, 'Experience Points').getAmount()
> 10)
```

The rule consequences (RHS) are maintained in the *Consequences ("then")* area. Examples are:

- Update achievements - “Experience Points” of the player are increased by 5:

```
updateAPI.givePoints($playerid, "Experience Points", 5, "Task done" );
update(engine.getPlayerById($playerid));
```

- Create new events - new event with the type “solvedProblemDelayed” that is triggered with a delay of 1 minute:

```
EventObject obj = new EventObject();
obj.setType("solvedProblemDelayed");
obj.setEventDuration(60*1000);
obj.setPlayerid($playerid);
entryPoints["unmanagedstream"].insert(obj);
```

> **i** Note
>
> The gamification service follows the “rule-first” approach. This means that any achievements of a player are always updated using the rule engine. A modification of player achievements cannot be done using an API (without any rule execution).

1.7.6.5.2 Rules Language

The Gamification allows you to write rules to reach the best flexibility for the targeted game concept. Additionally you can write rules in one of the multiple graphical (form based) editors in the gamification workbench.

1.7.6.5.2.1 Defining the Trigger (“WHEN”)

The declaration of the trigger (“when”) part is based on the Drools Rules Language (DRL).

The trigger part defines the constraints that must be fulfilled in order to execute the consequences ("then" part). Variables can be defined and used both in the "when" and in the "then" part. This is generally recommended in case you want to use the same object more than once. Multiple constraints can be described in one trigger part. The constraints are typically described using the logical operators (within eval statements) and evaluation of the event object. The event object must be defined with a type and can include multiple parameters. Additionally, DRL allows you to define temporal constraints using common complex event processing (CEP) operators.

Related Information

http://docs.jboss.org/drools/release/5.6.0.Final/drools-expert-docs/html/ch05.html

1.7.6.5.2.1.1 Entry Points (Event Streams)

The gamification service rule engine allows the use of two event streams:

- Managed event stream - `eventstream`: All events and user actions that are sent using the API will automatically be sent using the managed event stream. “Managed” means that all events are retracted automatically. Point-in-time events (duration=0) are retracted immediately after execution of the corresponding rules while long-living events (duration >0) are retracted 1 second after they have expired. If this automated event retraction is not suitable for your use case, you can use the unmanaged stream instead.
- Unmanaged event stream - `unmanagedstream`: For this stream you must take care of event retraction yourself, which offers more flexibility with regards to rule design. For stability reasons, events sent to this stream are retracted automatically after 28 days.

You must explicitly declare in the trigger part which event stream will be used. Furthermore, you must explicitly declare in the consequences part which event stream is used in case you create new events. Using the managed stream is strongly recommended. Only use the unmanaged stream if the auto-retraction does not work with your rule design.

1.7.6.5.2.1.2 Declaring Variables

Context

Variables can be defined in the trigger part and can afterwards be used in both the trigger and the consequences part. Variables are recommended in case one object is used more than once. For example, a player object needs to be updated multiple times.

Procedure

A variable is declared by any string with a leading `$` sign, for example `$player` or `$var`.

Declaration of a variable:

```
$<VARIABLE> : <EXPRESSION>
```

Examples for variable declaration:

- Binding of a variable for the player object.

  ```
  $player : Player($playerid : uid)
  ```

- Declaration of a variable for the event object.

  ```
  $event : EventObject(type=="solvedProblem", $playerid:playerid) from entry-
  point eventstream
  ```

1.7.6.5.2.1.3 Declaring Event Objects

Context

An event type must be set for each incoming event. The event type needs to be checked within the trigger part. The player's ID is sent with each event, it should be stored in a variable for further use.

Additionally, multiple parameters can be passed with an event and evaluated. The parameters can be a string or any numeric values. The parameters can be evaluated with logical operators such as equal (=), larger than (>) and smaller than (<).

Multiple evaluations of event objects can be triggered in one rule.

Procedure

Declaration of an event object with a given event type and declaration of a variable with a given player ID:

```
EventObject(type=='<EVENT_TYPE>', $playerid:playerid) from entry-point
eventstream
```

> **i Note**
>
> It is recommended to always assign the player ID (`playerid`) within the event object of a variable since the player ID is necessary to get the according player object for updating achievements in the consequence part.

Declaration of an event with a given event type, declaration of a variable with a given player ID and evaluation of a property:

```
EventObject(type=='<EVENT_TYPE>', data['<PROPERTY>']<OPERATOR><VALUE>
$playerid:playerid) from entry-point eventstream
```

> **i Note**
>
> It is recommended to always evaluate event parameters within the event object instead of defining additional parameters and using additional eval statements.

Examples for event declaration:

- Declaration of event with the given type “solvedProblem”.

  ```
  EventObject(type=='solvedProblem', $playerid:playerid) from entry-point
  eventstream
  ```

- Declaration of event with the given type “buttonPressed”.

  ```
  EventObject(type=='buttonPressed', $playerid:playerid) from entry-point
  eventstream
  ```

Examples for an event with given properties:

- Declaration of event with the given type “solvedProblem” and a property with the name “relevance” and the value “critical”.

```
EventObject(type=='solvedProblem', data['relevance']=='critical',
$playerid:playerid) from entry-point eventstream
```

- Declaration of event with the given type “buttonPressed” and a property with the name “color” and the value “red”.

```
EventObject(type=='buttonPressed', data['color']=='red', $playerid:playerid)
from entry-point eventstream
```

- Declaration of event with the given type “temperatureIncreased” and an integer property with the name “temperatureValue” where the numeric value is larger than 30.

```
EventObject(type=='temperatureIncreased',
Integer.parseInt(data['temperatureValue'])>30, $playerid:playerid) from entry-
point eventstream
```

Examples for the combination of event declarations:

- Declaration of two events of type “ticketEventA” and “ticketEventB”. Both events must occur and they have to belong to different players.

```
EventObject(type=='ticketEventA', $playerid:playerid)
EventObject(type=='ticketEventB', playerid!=$playerid)
```

- Declaration of two events of type “ticketEventA” and “ticketEventB” using the explicit “and” operator. Both events must occur and they have to belong to different players.

```
(EventObject(type=='ticketEventA', $playerid:playerid) &&
EventObject(type=='ticketEventB', playerid!=$playerid))
```

- Declaration of two events of type “ticketEventA” and “ticketEventB” using the “or” operator that describes that “eventA” or “eventB” must occur and the "player IDs" must not be the same.

```
(EventObject(type=='ticketEventA', $playerid:playerid) ||
EventObject(type=='ticketEventB', playerid!=$playerid))
```

- Declaration of two events of type “ticketEvent” where the “player IDs” are different and the “ticked id” is the same and another event of the type “connectedEvent” that must not be true.

```
EventObject(type=='ticketEvent', $ticketid:data['ticketid'],
$playerid:playerid) EventObject(type=='ticketEvent', data['ticketid']==
$ticketid, playerid!=$playerid, $playerid2:playerid)
not(EventObject(type=='connectedEvent', playerid==$playerid,
data['friendid']==$playerid2))
```

1.7.6.5.2.1.4 Declaring Eval Statements

Context

Eval statements are used to define constraints with data that is not available in the working memory, such as status of player achievements. Multiple constraints can be defined in one rule with the combination of multiple logical operators.

> **i Note**
>
> The code within eval statements must follow the Java syntax, just like in the case of the consequences ("then") part. They are not based on the Drools Rule Language like the rest of the trigger part.

> **i Note**
>
> It is recommended to avoid using an eval statement since it is an expensive operation. Use it as late as possible within your trigger part.

Procedure

Declaration of an eval statement:

```
eval(<EXPRESSION><OPERATOR><VALUE>)
```

- Expression: It is recommended to only use methods of the Query API in eval conditions. The use of the Query API allows you to evaluate available player details and achievements using Java statements.
- Operator: All logical operators supported by Java are supported.

Examples for eval statements:

- Declaration of an eval statement where the mission “Troubleshooting” is assigned to the player.

  ```
  eval(queryAPIv1.hasPlayerMission($playerid, 'Troubleshooting') == true
  ```

- Declaration of an eval statement where the “Experience Points” of the player are larger or equal to 10.

  ```
  eval(queryAPIv1. getScoreForPlayer($playerid, 'Experience Points ', null,
  null).getAmount() >= 10)
  ```

- Declaration of an eval statement where the player does not have the badge “Sporting Ace” assigned.

  ```
  eval(queryAPIv1.hasPlayerBadge($playerid, 'Sporting Ace') == false)
  ```

> **i Note**
>
> The use of an invalid expressions may lead to an error during rule execution. Make sure that referenced point categories or missions exist and the spelling is correct.

1.7.6.5.2.1.5 Using Generic Facts

Creating generic facts (a Map object with an optional key) and storing them in the working memory is supported. This allows you to store temporary results and create complex constraints (e.g.: count the number of a specific event type). Generic facts can be evaluated in all rules if they exist.

The data structure of a generic fact is Map<String, Object> data. Additionally, you can set a key for the generic factr to identify it. A generic fact must be initialized in the consequences part.

In the trigger ("when") part you typically query certain facts from the memory using the following syntax:

```
GenericFact(key=='<KEY>')
```

Declaration of a variable with a generic fact value:

```
$<FACT_VARIABLE>: GenericFact(key=='<KEY>')
```

Examples for querying generic facts and assignment to a variable that can be used for evaluation:

- `$loginCounter: GenericFact(key=='LoginCounter')`

  Declaration of a generic fact “loginCounter”.
- `$daysOfWeek: GenericFact(key=='DaysOfWeek')`

  Declaration of a generic fact “daysOfWeek”.

1.7.6.5.2.2 Defining the Consequences ("THEN")

The declaration of the consequences (“then”) part supports writing code with the Drools Rules Language (DRL) in version 5.6.0 and Java code.

> **i Note**
>
> The formatting in the consequences part must be in the Java style. The DRL can be used in combination with Java code.

The consequences part defines what will be executed once the trigger part is fulfilled. It allows you to update the player achievements or to create new events. Multiple consequences can be defined within one consequences part.

Related Information

http://docs.jboss.org/drools/release/5.6.0.Final/drools-expert-docs/html/ch05.html

1.7.6.5.2.2.1 Updating Player Achievements

The Update API can be used to update any player achievements. Multiple updated can be executed within on the consequences part.

Declaration of an update on a player achievement:

```
updateAPIv1.<QUERY_API_METHOD>(<PLAYER_ID>, <PARAMS>);
update(engine.getPlayerById(<PLAYER_ID>));
```

The player ID has to be declared in the trigger part. Once all updates are executed the update method with the player ID has to be executed once.

Examples for player achievement update declaration:

- Increasing the “Experience Points” of the player by one.

```
updateAPIv1.givePoints($playerid, 'Experience Points', 1, 'Task completed.');
update(engine.getPlayerById($playerid));
```

- Assign the mission “Troubleshooting” to the player.

```
updateAPIv1.addMissionToPlayer($playerid, 'Troubleshooting');
update(engine.getPlayerById($playerid));
```

- Player completed the mission “Troubleshooting”.

```
updateAPIv1.completeMission($playerid, 'Troubleshooting');
update(engine.getPlayerById($playerid));
```

- Assign the badge “Champion Badge” to the player.

```
updateAPIv1.addBadgeToPlayer($playerid, 'Champion Badge', '1. place');
update(engine.getPlayerById($playerid));
```

Examples for multiple player achievement update declaration:

- Increasing the “Experience Points” of the player by one, complete mission “Troubleshooting, and add badge “Champion Badge”.

```
updateAPIv1.givePoints($playerid, 'Experience Points', 1, 'Task completed.');
updateAPIv1.completeMission ($playerid, 'Troubleshooting');
updateAPIv1.addBadgeToPlayer($playerid, 'Champion Badge', '1. place');
update(engine.getPlayerById($playerid));
```

1.7.6.5.2.2.2 Creating New Events

New events can be created in the consequences part. They can be used for more complex game mechanics (cascading rules), changing the state of facts or even for temporal triggers.

Declaration of a new event:

The player ID has to be declared in the trigger part.

```
EventObject obj = new EventObject();
obj.setType(<TYPE>);
obj.setPlayerid(<PLAYER_ID>);
entryPoints["eventstream"].insert(obj);
```

Examples for creating new events:

- New event of type “solvedProblemCriticalTwice”.

```
EventObject obj = new EventObject();
obj.setType("solvedProblemTwice");
obj.setPlayerid($playerid);
entryPoints["eventstream"].insert(obj);
```

- New event of type “solvedProblemCriticalTwice” with a parameter “relevance”.

```
EventObject obj = new EventObject();
obj.setType("solvedProblemTwice");
obj.setPlayerid($playerid);
obj.put("relevance", "critical");
entryPoints["eventstream"].insert(obj);
```

Examples for creating new events which is executed delayed:

- New event of type “solvedProblemDelayed” which is executed with a delay of 1 hour.

```
EventObject obj = new EventObject();
obj.setType("solvedProblemTwice");
obj.setPlayerid($playerid);
obj.setEventDuration(60 *60*1000);
entryPoints["eventstream"].insert(obj);
```

1.7.6.5.2.2.3 Working with Generic Facts (Storing Arbitrary States)

Generic facts can be used as global variables and are stored in the working memory. The creation of a generic fact instance has to be done in the consequences part. In the trigger part you can query for certain generic fact instances and (if required) bind them to local variables. This works just like querying the `EventObject`.

Creation of a generic fact:

Creation of a generic fact with a given key.

```
GenericFact myFact = new GenericFact(<KEY>);
insert(myFact);
```

Examples for the creation of generic facts:

- Declaration of a generic fact with the key “factA”.

```
GenericFact factA = new GenericFact('factA');
insert(factA);
```

- Declaration of a generic fact with the key “factB” with a property “relevance” and according value “critical”.

```
GenericFact factB = new GenericFact('factB');
factB.addData('relevance', 'critical');
insert(factB);
```

Reading a generic fact:

The generic fact variable has to be declared in the trigger part.

```
$<FACT_VARIABLE>.getData();
```

Examples for reading generic facts:

- Assigning the value of the generic fact “loginCounter” to a variable.

```
int lCounter = $loginCounter.getData();
```

- Assigning the data of the generic fact “daysOfWeek” to a variable of type HashMap.

```
HashMap<String,Object> hmDays = $daysOfWeek.getData();
```

Updating a generic fact:

The generic fact variable has to be declared in the trigger part.

```
$<FACT_VARIABLE>.setData(<VALUE>);
update($<FACT_VARIABLE>);
```

Examples for updating generic facts:

- Assigning the value “59” to the generic fact “loginCounter”.

```
$loginCounter.setData("59");
update($loginCounter);
```

- Assigning the value of the variable “lCounter” to the generic fact “loginCounter”.

```
$loginCounter.setData(lCounter);
update($loginCounter);
```

Deleting a generic fact:

The generic fact variable has to be declared in the trigger part.

```
retract($<FACT_VARIABLE>);
```

Examples for deleting generic facts:

```
retract($loginCounter);
```

Examples for querying or binding generic facts in WHEN-part:

- ```
  $event : EventObject(type=='wpbStartChapter', $playerid:playerid,
  $chapter:data['chapterId']) from entry-point eventstream
  $p : Player($playerid == uid)
  not GenericFact(key==$playerid,data['chapterId']==$chapter)
  ```

1.7.6.5.2.2.4 Advanced Consequences

Using Java code in the consequences part is allowed and very complex rules can be created. You can work with all Java control flow statements, a selected set of Java objects, for example collections, create generic facts or update the player's achievements.

Example code for advanced consequences:

- Getting the day of the week.

```
java.text.SimpleDateFormat df = new java.text.SimpleDateFormat('d/MM/yy');
df.applyPattern("F");
java.util.Date _date = new Date();
int day=Integer.parseInt(df.format(_date));
```
```

1.7.6.5.3 Create Rules

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Rules* tab.

Procedure

1. Create a rule context.
   a. Press *Add* in the rules entity list to add a new rule.
   b. Enter a name.
   c. Optional: Enter a display name.
   d. Optional: Enter a description.
   e. Optional: Enter the priority (Rules with lower number have higher priority.)
   f. Press *Create*.
2. Enter the Rule logic (rules language).
   a. Enter rule code for trigger in the *Condition* window. The trigger code describes when a rule shall become valid.
   b. Enter rule code in the *Consequences* window. The consequence code describes what shall happen once a rule becomes valid.
   c. Optional: Select *Activate on Engine Update* if the rule should become enabled after applying the rule changes. (Value is selected by default.)
   d. Press *Save*.
   e. Optional: Create or modify additional rules.

> ⚠ Caution
>
> A newly created rule is not automatically deployed. The deployment is initiated once you apply the changes. The rule must be activated to be deployed.

1.7.6.5.4 Enable Rules

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Rules* tab. A rule already exists and is not enabled.

Procedure

1. Check the *Activate on Engine Update* checkbox of the rule you want to enable.
2. Open the Rule Engine Manager by pressing *Rule Engine*.
3. Commit your changes by pressing the *Apply Changes* button in the *Rule Engine Manager*. The rule will be deployed immediately after successful validation. A blue flag next to the rule indicates that the rule has been changed.

> **i** Note
>
> A rule that contains errors will not be deployed. Errors can be viewed by pressing the *Show Issues* button in the *Rule Engine Manager*.

1.7.6.5.5 Disable Rules

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Rules* tab. A rule already exists and is enabled.

Procedure

1. Uncheck the *Activate on Engine Update* checkbox of the rule you want to disable.
2. Open the Rule Engine Manager by pressing *Rule Engine*.
3. Commit your changes by pressing the *Apply Changes* button in the *Rule Engine Manager*. The rule will be deployed immediately once the validation was successful.

1.7.6.5.6 Update Rules

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Rules* tab.

Procedure

1. Click on the name of the rule in the entity list to open the rule editor.
2. Change the rule code.
3. Press *Save*.
4. Optional: Create or modify additional rules.
5. Close the rule editor and apply changes to deploy the rules.

> ⚠ Caution
>
> A modified rule is not automatically deployed. The deployment is initiated once you have pressed *Apply Changes* in the rules overview. The rule must be enabled to be deployed.

1.7.6.5.7 Delete Rules

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Rules* tab.

Procedure

1. Select the rule in the entity list to be deleted.
2. Press *Delete*.
3. Confirm deletion by pressing *Confirm* in the popup dialog.

> ⚠ Caution
>
> Only rules that are disabled can be deleted.

1.7.6.5.8 Detect Errors

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Rule Engine* tab.

Context

The gamification workbench supports to detect issues with rules during design time and during runtime. Any detected issues will be displayed in the *Rule Engine* tab. Syntax errors are already checked during design time after the user applied the changes.

Procedure

1. Reported rule warnings are displayed in a table, sorted by the rule which caused them.
2. Optional: Press the refresh button attached to the rule warnings table to refresh and check for new warnings.

1.7.6.5.9 Recover Rule Snapshots

Prerequisites

You have logged on to the gamification workbench with the role `TenantOperator` or `AppAdmin`, and you have opened the *Rule Engine* tab of the releated app.

Context

The gamification service creates a rule engine instance for each app. Over time the state of each rule engine instance changes based on its usage. A recovery mechanism for different rule engine states has been introduced to allow a clean recovery in case of errors, rule set changes or system migrations. This mechanism allows to create and restore snapshots of the current rule engine instance session and its deployed rule set. Snapshots are stored into the database.

Generation of snapshots

Using “apply changes” (see Update Rules [page 670] for details), the current rule set stored in the database is deployed on the currently running rule engine instance. Technically, the current session, which includes all facts and events, is upgraded to a new rule set. To assure compatibility of new rules with the existing session, rules are being evaluated one by one. Compatible pairs of session and rule set are stored as snapshots.

Additionally, when receiving events via the “handleEvent” method, the session will change as well and requires the same recovery mechanism. The gamification service service will generate snapshots during event execution in dynamic intervals.

The gamification service manages rules and corresponding snapshots in the following way:

- After each successful rule deployment (*Apply Changes*) the corresponding rule set as well as the session are both tagged with a new version. The service stores the latest 10 versions at max.
- For the latest (currently active) version as well as the previous version the gamification service stores the 10 latest snapshots in slots numbered 1 through 10.

- By using the API and the Workbench you can retrieve all available snapshots as well as the corresponding rule set. Additionally, the rule engine can be restored to any of these snapshots.

Procedure

1. The *Rule Engine* section lists a table with all available rule engine snapshots and their details.
2. Choose a rule engine snapshot to recover and press its *Recover* button.
3. Read and confirm the modal dialog.
4. The gamification service is now recovering the snapshot. This may take a few seconds.

> **i** Note
>
> Rule engine snapshots are constantly being created, when events are being sent. Older snapshots are removed by the system during the process. It is recommended to stop any applications from sending events to the rule engine while restoring snapshots.

Related Information

Event Submission API documentation
Game Rules Management API documentation

1.7.6.6 Using the Notification Mechanism

Notifications are messages that inform users about certain state changes, for example earned achievements, new missions, new teams. They are considered "see and forget" information and won't stay long in the system.

Context

On one hand, notifications are created automatically when calling certain API methods. On the other hand, you can also create and assign custom notifications by using the methods `addCustomNotificationToPlayer` and `addCustomNotificationToTeamMembers`.

Notifications are delivered to players or teams by implementing a polling-based approach using the API methods `getNotificationsForPlayer` and `getAllNotifications`.

1.7.6.6.1 Automatic Notifications

The gamification service automatically creates notification for users when calling certain API methods. The table below lists all methods, which implicitly generate notifications and explains the corresponding notification parameters.

API Method	Player	Type	Category	Subject	Details	Message	Date Created
addBadgeToPlayer	id	ADD	BADGE	Badge Name	-	custom	timestamp
givePoints	id	ADD	BADGE	Point Name	amount	custom	timestamp
addMissionToPlayer	id	ADD	MISSION	Mission Name	-	custom	timestamp
completeMission	id	COMPLETE	MISSION	Mission Name	-	custom	timestamp
addPlayerToTeam	id	ADD	TEAM	Team Name	Player Name	custom	timestamp
deletePlayerFromTeam	id	REMOVE	TEAM	Team Name	Player Name	custom	timestamp
addMissionToTeam	id	ADD	TEAMMISSION	Mission Name	Team Name	custom	timestamp
completeTeamMission	id	COMPLETE	TEAMMISSION	Mission Name	Team Name	custom	timestamp
givePointsToTeam	id	ADD	TEAM	Point Name	amount	custom	timestamp

Custom messages can usually be specified using an optional parameter `<notificationMessage>` of the corresponding API method.

Examples:

- `addBadgeToPlayer(String playerId, String badgeName, String <notificationMessage>)`
- `givePointsToPlayer(String playerId, String pointType, double amount, String <notificationMessage>)`

1.7.6.6.2 Custom Notifications

Besides the automatically generated notification it is possible to add custom notifications to players or teams using the methods `addCustomNotificationToPlayer` and `addCustomNotificationToTeamMembers` from within rules.

The table explains how the notification parameters are used when creating custom notifications.

API Method	Player	Type	Category	Subject	Detail	Message	Date Created
addCustom-Notification-ToPlayer	id	CUSTOM	Any	optional	-	custom	timestamp
addCustom-Notification-ToTeamMem-bers	id	CUSTOM	Any	optional	-	custom	timestamp

1.7.6.6.3 Consume Notifications

Context

Notifications are strictly defined as "see and forget". The gamification service will only store the last 25 notifications for each player (currently "X" defaults to 25). The show notifications to players a polling-based approach has to be implemented using the following API methods:

- `getNotificationsForPlayer(playerId, timestamp)&app=APPNAME`
  Returns the latest notifications for a player starting from the timestamp. This mechanism allows other applications to better track which notifications have been requested or displayed already. This is the current approach for "user2service" communication. It works well with the user endpoint using JavaScript.
- `getAllNotifications(timestamp)&app=APPNAME`
  Returns all generated notifications for all players within one app starting from the provided timestamp. This is the current approach for "application2service" communication. An application can query all notifications for the app using the tech endpoint and forward the information to the user using custom events or communication channels. This avoids having all clients in parallel polling for notifications.

Procedure

You can see the Notification Widget in the Helpdesk Scenario (sap_gs_notifications.js) for more information on how the polling of notifications can be implemented at the client side. The notification polling is handled as follows:

1. Retrieve the gamification service server time on initialization, using the method `getServerTime`.

2. Use this server time to initially poll for notifications.

   For example `getNotificationsForPlayer(playerId, servertime)` or `getAllNotifications(servertime)`.

3. When new notifications are received from the service, replace the polling timestamp with the `dateCreated` timestamp from the youngest notification received.

   For example `getNotificationsForPlayer(playerId, youngestNotificationDateCreated)` or `getAllNotifications(youngestNotificationDateCreated)`.

1.7.6.7 Execute API calls

Prerequisites

You have logged into the gamification workbench and opened the *Terminal* tab.

You have the role `GamificationDesigner`.

Context

The Terminal within the game mechanics area allows you to quickly execute one or more API calls. Make sure that you have the appropriate access rights for executing the call.

A comprehensive documentation of the API can be found in your Gamification subscription under *Help* > *API Documentation*.

Procedure

1. Enter the list of JSON RPC calls as a JSON array: [JSON_RPC_CALL1, JSON_RPC_CALL2,...]

   Example:

   ```
   [{"method":"createMission", "id":1, "params":["missionname", "mission
   description", "mission consequence"]},{"method":"createLevel", "id":1,
   "params":["name","1","Experience Points"]}]
   ```

2. Press *Execute* to execute the calls. Check *Force synchronous execution* checkbox to enforce sequential execution of calls in the JSON array.
3. Review server response. You can view the detailed JSON response by clicking on the symbol on the right.

> **i Note**
>
> The calls are executed in the context of the currently selected app (see dropdown box in the upper right corner of the gamification workbench).

The defined JSON RPC calls are stored in the browser cache. For restoring the initial sample calls press *Restore Example*.

Press the *Restore Example* button in the *Terminal* section to show some example requests. Use the API Documentation (*Help* > *Open API Documentation*) to find a list of all available methods.

Related Information

Send Player Events [page 677]
Get Player Achievements [page 678]

1.7.6.7.1 Send Player Events

Prerequisites

Navigate to the Terminal in the *Game Design* tab. Your user has the role `AppAdmin`.

Context

The Terminal allows you to send events that are typically sent to the host application.

> **i Note**
>
> The Terminal should be only used to send events for testing purposes. In case you send events for a user that is used in a productive environment it will modify the real achievements!

Procedure

1. Enter the list of JSON RPC calls with the method `handleEvent`.

```
[ {"method":"handleEvent", "params":[{"type":"myEvent","playerid":"demo-user@mail.com","data":{}}]} ]
```

2. Press *Execute* to execute the calls. Check *Force synchronous execution* checkbox to enforce sequential execution of calls in a JSON array.
3. Review server response. You can view the detailed JSON response by clicking on the symbol on the right. Once the event is send successfully the response is `true`.
4. All rules that listen on the according event type (when clause) will be executed.

1.7.6.7.2 Get Player Achievements

Prerequisites

Navigate to the Terminal in the *Game Mechanics* tab.

Context

The Terminal allows you to execute all methods for retrieving the user achievements data.

Procedure

1. Enter list of JSON RPC calls with the method with the desired achievement query methods.

   Example `getPlayerRecord`:

   ```
   [ {"method":"getPlayerRecord", "params":["demo-user@mail.com"]} ]
   ```

2. Press *Execute* to execute the calls. Check *Force synchronous execution* checkbox to enforce sequential execution of calls in a JSON array.
3. Review server response. You can view the detailed JSON response by clicking on the symbol on the right. Once the event is send successfully you will see the result.

1.7.6.8 Search Event Logs

Prerequisites

You are logged into the gamification workbench and have opened the *Logging* tab.

You have the role TenantOperator.

Context

The logging view allows you to search the event log for the selected app. The event log includes all API calls related to “Event Submission” as well as the corresponding API calls executed from within the rules, which were triggered by the corresponding events.

Procedure

1. View event log for most recent entries, covering:
   - Event Submission API calls, for example `handleEvent` or `applyChanges`
   - All API calls executed within triggered rules
2. (Optional) Configure the filter for the event log. Supported filtering options:
   - Event Submission API calls, for example `handleEvent` or `applyChanges`
   - All API calls executed within triggered rules
3. (Optional) Press *Go* to update the event log using the specified filter.

> **i Note**
>
> The maximum retention time for the event log is 7 days, but not exceeding 500,000 log entries.

1.7.6.9 Troubleshooting Guide

Rules with an `EventObject` fact and one or more other facts (`Player` or `GenericFact`) in WHEN part cause endless loops.

Understanding why such rule sets result in loops requires a deeper understanding of the gamification service itself:

- Rules with fact-based conditions are triggered on changes of the respective fact or facts. For example, insert, update or retract fact.
- `handleEvent` inserts a fact of type `EventObject` and fires all rules. For example the THEN parts of all rules that satisfy a fact-based condition involving `EventObject` will be executed.
- THEN execution may involve the modification of facts (insert, update, delete), which in turn may trigger further rules. For example, insert a new `GenericFact` or update an existing fact (`Player` or `GenericFact`). Rule execution runs until there are no more rules to fire.
- Endless loops occur if there are circles in the rule execution graph, for example, one rule calling another and vice versa. The gamification service loop detection will detect such loops at runtime and stop the engine until the problems are resolved.
- The `EventObject` inserted by `handleEvent` is per default retracted automatically after all rules have fired. Thus, if the WHEN part includes `EventObject` conditions and further fact conditions, for example, Player(), the rule will trigger again if one of the respective facts changed and the overall condition is still true.
- This can cause an endless loop. For example: Rule 1 WHEN includes `EventObject` and queries for corresponding player (`Player(playerid==$playerid)`). Rule 2 WHEN expects `Player` change only (`Player()`) in WHEN. If both, Rule 1 and Rule 2, include an update(`$player`) in the THEN part, this will result in an endless loop.

Mitigation strategy

- Use `update(fact)` with care. Think if it is needed and check for rules that could trigger accidently.
- Minimize the number of update calls in the THEN part. Example: Only call update($player) if player achievement data has changed and you want other rules to retrigger, e.g. rules checking for mission

completion. This will also significantly improve performance since unnecessary rule executions are avoided.

- Consider using predefined fact query methods in the THEN part if sufficient. For example:
  - `engine.getGenericFactByKey($playerid);`
  - `engine.getPlayerById($playerid);`
- Example for accessing `GenericFact` in THEN;
  - `DefaultFactHandle playerFactHandle = engine.getGenericFactByKey($playerid);`
  - `String data = ((GenericFact)playerFactHandle.getObject()).getData().get('playerid');`

Class cast exceptions when working with number-based conditions on `EventObject` key/value data.

Both, key and value are interpreted as Strings. Thus, an explicit type conversion is required if you want to compare them with numbers. This type conversion is done using the standard Java approach for the different numeric types, for example, `Integer.parseInt(value)` or `Double.parseDoube(value)`.

Example:

- WHEN: `$event : EventObject(type=='solvedProblem', Integer.parseInt(data['processTime']) < 20, $playerid:playerid)` from entry-point eventstream
- Event (via Terminal):

```
[
{"method":"handleEvent", "params":
[{"type":"solvedProblem","playerid":"D053659","data":
{"relevance":"critical","processTime":15}}]}
]
```

- Binding variables for further use:
  `$event : EventObject(type=='solvedProblem', $pt:Integer.parseInt(data['processTime']))` from entry-point eventstream

Related Information

Rules Language [page 660]

http://help.sap.com/disclaimer?site=http://docs.jboss.org/drools/release/5.6.0.Final/drools-expert-docs/html/ch05.html

1.7.7 Integrating Gamification into a Target Application

Context

The integration of a (gamified) cloud application must consider the following aspects:

1. Sending gamification-relevant events to a player or a team, for example the user has completed a task for which the gamification service grants a point.
2. Giving feedback to the players/teams, for example by showing achievements, progress, and game notifications, .
3. Integrating the user management - creating or enabling players/teams, blocking players/teams, deleting players/teams.

The following sections describe how you can deal with these aspects using the Web APIs provided. The sample code shown is based on the demo application "Help Desk". The demo application's source code is also available in GitHub .

> **i Note**
>
> The sample code used to demonstrate the integration is not ready for production.

1.7.7.1 Gamification API

The Application Programming Interface (API) of the gamification service is the central integration point of your application.

The gamification service API comprises two endpoints:

- *Technical endpoint* for integrating gamification events and user management in youur backend.
- *User endpoint* for integrating user achievements in the application frontend.

It is recommended to use the technical endpoint only for executing methods of the gamification service that must not be executed by the users themselves, such as sending events to the gamification service that trigger certain achievements or performing user management tasks, creating players for example. Authentication and authorization in this case is based on a technical user that is created for the application itself.

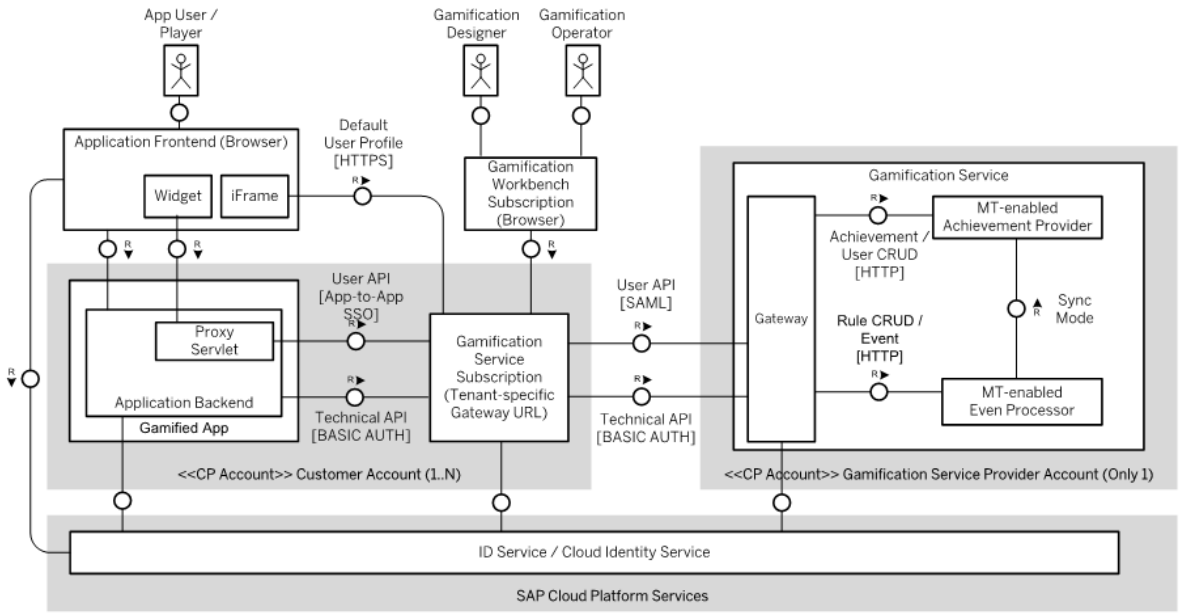
The user endpoint should be used for accessing user related information for example earned achievements, available achievements/mission, notifications and others. A great advantage of this approach is that the gamification service manages access control, based on the user roles. For instance to make sure that a user cannot access other users' data. For this, the authenticated user must be passed to the user endpoint.

> **i Note**
>
> The whole integration can be done by using only the technical endpoint. However, in this case you must manage access control yourself..

The documentation for the API can be found in your gamification service under *Help* > *API Documentation* or at https://gamification.hana.ondemand.com/gamification/documentation/documentation.html.

1.7.7.2 Integration Architecture Overview

The graphic below illustrates how a gamified application (gamified app) running on SAP BTP is typically integrated with the gamification service. The demo application "Help Desk" follows this integration architecture:



In a SAP BTP setting we assume that the gamified app and the gamification service subscription are located in the same subaccount. Furthermore, we assume that the application back end is written in Java, while the application front end is based on HTML5 or SAPUI5.

The technical endpoint is used to send gamification-relevant events and perform user management tasks from the application back end. Communication is based on a BASIC AUTH destination that uses the user name and password of a technical user.

> **i Note**
>
> For productive settings the client-side event sending should support resending events in case of failures, planned or unplanned service downtimes. For instance, short planned downtimes (less than 5 minutes according to Cloud Platform maintenance schedules) are required to apply regular gamification service updates.

The easiest way to show player achievements is to integrate a default user profile that comes with the gamification service subscription as an iFrame in the application's web front end.

To implement a user profile or single widgets (for example a progress bar tailored to the application's front end), we recommend you use the user endpoint in combination with a local proxy servlet and an app-to-app SSO destination. The proxy servlet prevents running into cross-site scripting issues and the app-to-app SSO destination automatically forwards the credentials of the authenticated user to the gamification service. This allows reuse of the access control mechanisms offered by the gamification service.

Since the user endpoint is used from a browser it is protected against cross-site request forgery. Accordingly, an XSRF token has to be acquired by the client first.

Related Information

Exchanging Data via HTTP [page 146]
Application-to-Application SSO Authentication [page 104]
Security Development [page 1677]

1.7.7.3 Send Gamification Events

Context

If the user performs actions in the application that are relevant to gamification, the gamification service has to be informed by invoking the corresponding API method. To prevent cheating this should be done in the application back end using the technical endpoint offered by the API.

> **i Note**
>
> For productive settings the client-side event sending should support resending events in case of failures, planned or unplanned service downtimes. For instance, short planned downtimes (less than 5 minutes according to Cloud Platform maintenance schedules) are required to apply regular gamification service updates.

Procedure

1. Create a destination to the technical endpoint
   a. Type: “HTTP”.
   b. URL: https://<Subscription URL>/gamification/api/tech/JsonRPC .
   c. Authentication: “Basic Authentication”.
   d. User: Technical user ID. The technical user must have roles `AppStandard` and `AppAdmin`.
   e. Password: The technical user's password.
2. Invoke method "handleEvent" in the appropriate place in the back end, for example after user action has been executed.

```
http:// <Subscription URL>/gamification/api/tech/JsonRPC?
json={"method":"handleEvent","id":1,"params":
[{"siteId":"HelpDesk","type":"solvedProblem","playerid":"abc@abc.com"}]}
```

> **i Note**
>
> See also:
> - Demo application source code: https://github.com/SAP/gamification-demo-app
> - API Documentation: Gamification subscription, under *Help* > *API Documentation*.

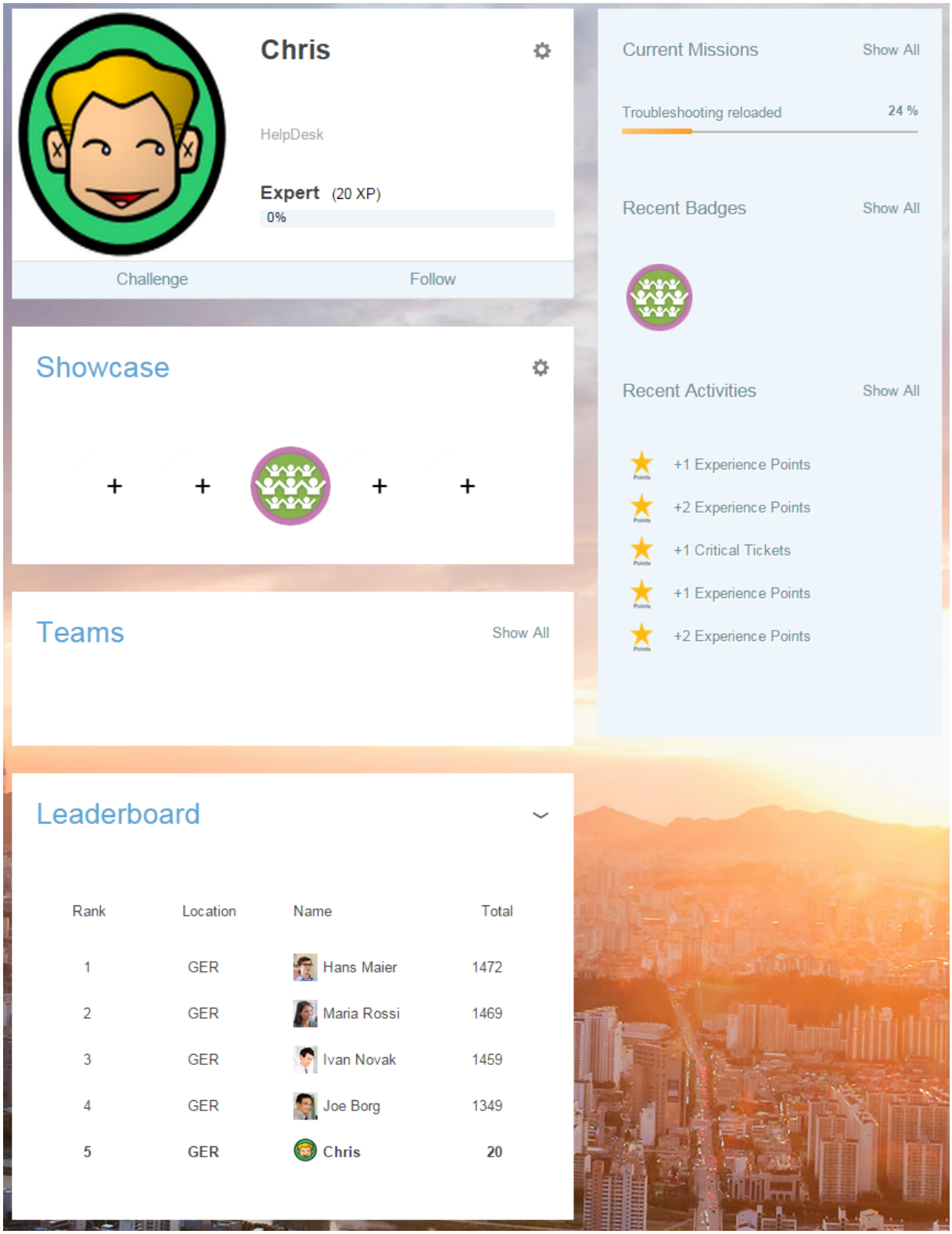
Related Information

Exchanging Data via HTTP [page 146]

1.7.7.4 Include the Default User Profile

Context

The gamification service subscription includes a default user profile, which you can include in your application as an <iFrame/>.


Chris
HelpDesk
Expert (20 XP)
0%
Challenge
Follow
Showcase
Teams
Show All
Leaderboard
Rank Location Name Total
1 GER Hans Maier 1472
2 GER Maria Rossi 1469
3 GER Ivan Novak 1459
4 GER Joe Borg 1349
5 GER Chris 20
Current Missions
Show All
Troubleshooting reloaded
24 %
Recent Badges
Show All
Recent Activities
Show All
+1 Experience Points
+2 Experience Points
+1 Critical Tickets
+1 Experience Points
+2 Experience Points

Procedure

1. See the default user profile using the following URL:

   ```
   https://<Subscription URL>/gamification/userprofile.html?
   name=<userid>&app=<appid>
   ```

2. Include the default user profile in your HTML5 code as an iFrame:

   ```
   <iframe src="https://<Subscription URL>/gamification/ userprofile.html?
   name=<userid>&amp;app=<appid>" width="100%" height="100%"
   frameborder="0" data-sap-ui-preserve="iframeProfile"
   id="iframeProfile">Alternate text if the iframe cannot be rendered</iframe>
   ```

1.7.7.5 Integrate Custom Widgets

Prerequisites

Configure your subaccount to allow principal propagation. For more information, see HTTP Destinations [page 91]

Context

The integration of custom gamification elements tailored to your application's user interface requires the development of custom JavaScript/HTML5 widgets. To avoid cross-site-scripting issues, you should introduce a proxy servlet in the application. This servlet forwards JSON-RPC requests to the user endpoint using an App-to-App SSO destination. This way, the gamification service has access to the user principle and the built-in access control is active.

Procedure

1. Configure your subaccount to allow principal propagation.
2. Create a destination to user endpoint.
   a. Type: “HTTP”.
   b. URL: https://<Subscription URL>/gamification/api/tech/JsonRPC .
   c. Authentication: “AppToAppSSO”.
3. Include a proxy servlet that receives JSON-RPC strings and forwards them to the user endpoint using the previously created destination. Refer to the API documentation for a list of available methods.
4. Include a JavaScript/HTML5 widget that sends JSON-RPC requests to the proxy servlet.

   API Documentation: Gamification subscription under *Help* > *API Documentation*.

   Demo application source code: https://github.com/SAP/gamification-demo-app

Related Information

Application-to-Application SSO Authentication [page 104]
Protection from Cross-Site Request Forgery [page 1840]

1.7.7.6 Integrate Player (User) Management

Context

The players (users) must be explicitly created before they can be used to assign achievements. A player context is always valid for one tenant and therefore can be used across multiple apps (managed in one tenant).

Procedure

1. Register (create) a player (user) for a tenant subscription using the API method `createPlayer`.

   > **i Note**
   >
   > This is done automatically on the first event if the flag Auto-Create Players is set to `true` for the given app.

2. (Optional) Initialize a player (user) by creating a rule listening for an event of type `initPlayerForApp`.
   a. Precondition: The player is registered.
   b. On event: if a player has not been initialized for the given app yet an event of type `initPlayerForApp` is automatically inserted into the engine. The THEN-part of this rule should include the user-defined init actions, for example assigning initial missions.
   c. (Optional) If you want players to be created with a display name you can add the optional parameter `playerName` to the event. During the automated player creation this parameter is used for setting the player name. Example:

   ```
   {"method":"handleEvent","params":
   [{"type":"linkProvided","playerid":"maria.rossi@sap.com", "playerName":
   "Maria Rossi", "data":{}}]}
   ```

3. Submit any events using the API method `handleEvent`.
   a. Precondition: The player is registered. Otherwise the call is rejected.
   b. According rules are triggered.
4. Optional: Remove a player from the app using the event `removePlayerFromApp`.
   a. Precondition: The player is registered and has been initialized (if necessary).
   b. The event has to be triggered explicitly by the host application.
   c. A rule has to be provided that removes the player fact and performs all further scenario-specific clean-up actions.

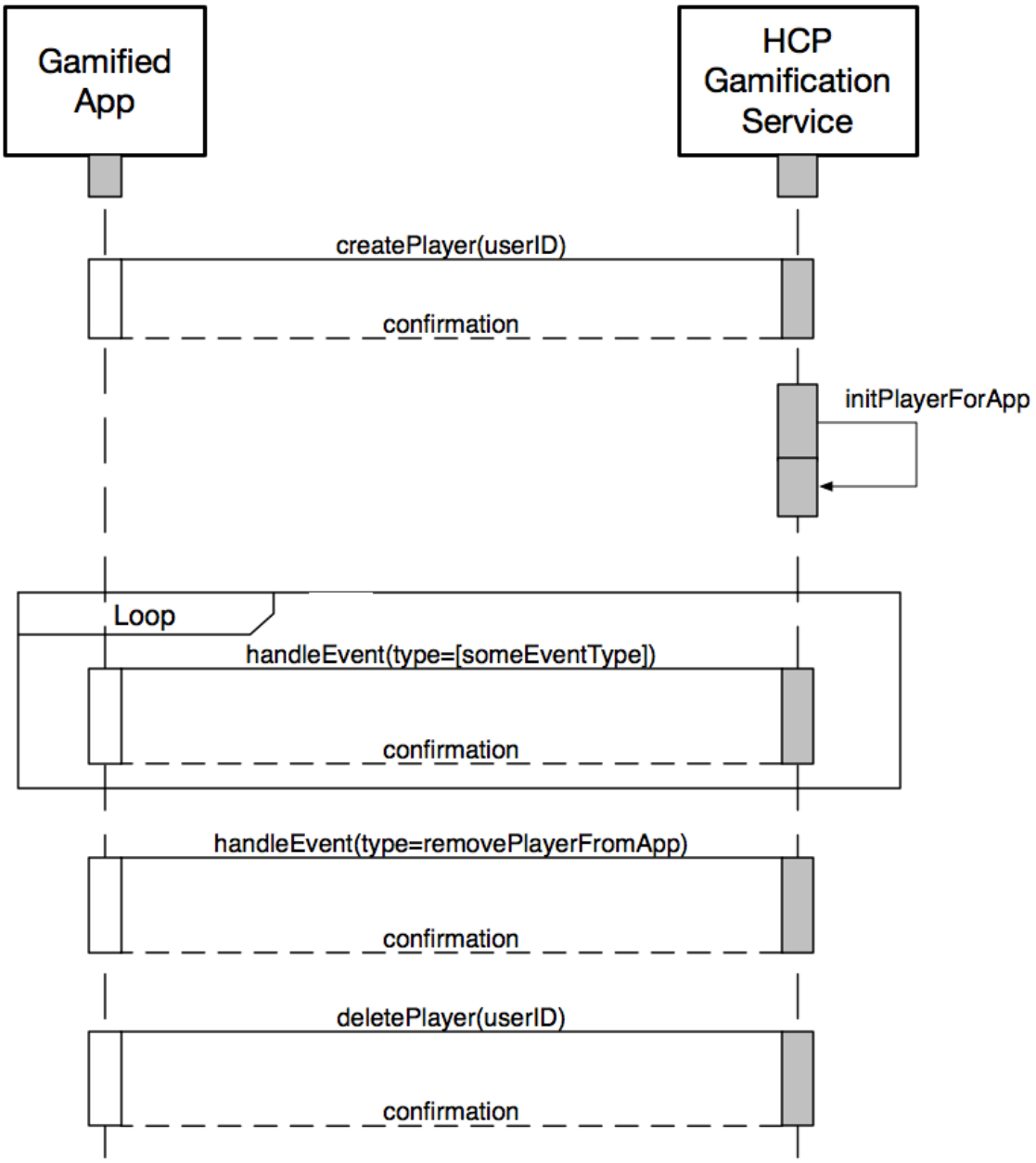
Sample minimal clean-up rule:

- WHEN-Part: Select user fact based on id provided with removePlayerForApp event

```
$event : EventObject(type=='removePlayerFromApp', $playerid:playerid)
from entry-point eventstream
$p : Player(id==$playerid)
```

- THEN-Part: Retract the given player fact
  retract($p);

5. Delete player (user) for tenant subscription using the API method `deletePlayer`.



1.7.8 Analyzing Gamification Concepts

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Game Design* tab.

Context

The gamification introduction is a continuous process since the modification of game mechanics can be done at any point in time. For example, the number of points a player can reach might be changed in order to change the behavior of the user.

The analytics can be executed in the *Analytics* tab.

1.7.8.1 View Statistics of Points Metric

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Analytics* tab.

Context

You can view the statistics of achievements such as points and badges. The points metrics that can be viewed are all point categories and badges that are maintained for your application.

The following aggregations can be selected (the values for badges cannot be aggregated):

- Max - The maximum of the selected values.
- Sum - The sum of the selected values.
- Avg - The average of the selected values.
- Count - The number of occurrences of the selected values.

The values can be grouped by time or other values:

- Day - Group by day.
- Month - Group by month.
- Year - Group by year.
- Team - Group by team.
- Badge - Group by badge.
- Level - Group by level of players.

The data can be filtered for a user defined time range (if no time range is selected, all data is displayed):

- From - The minimum value of the time range.
- To - The maximum value of the time range.

The selected values will be displayed as a bar chart.

> **i Note**
>
> The analytics are currently limited to point categories and badges. Analytics on player level are not available due to privacy reasons.

Procedure

View statistics for a point metric:

1. Select the points metric.
2. Optional: Select the aggregation.
3. Optional: Select the group by filter.
4. Optional: Check the *Time range* checkbox.
5. Optional: Select the time range (from / to) in case you have selected the *Time range* checkbox.

1.7.8.2 View Difference in Values to Earlier Time Range

Prerequisites

You have logged on to the gamification workbench with the role `GamificationDesigner` and you have opened the *Analytics* tab. You have selected the statistics you are interested in. A time range must be selected.

Context

You can view the statistics of achievements such as points and badges. The selected values can be compared to an earlier time range in order to identify changes in the assignment of achievements.

> **i Note**
>
> A time range for the statistics must be selected.

Procedure

View a lag chart for a comparison of the selected data to an earlier time range.

1. Select the *Enable lag chart* checkbox.
2. Select the lag amount for comparison.

   The lag chart displays the difference of the aggregated values to the values before the lag amount. For example, when you select the sum of point category for the current month, the lag chart will show the difference compared to the month before, provided you have selected a lag amount equal to one month.

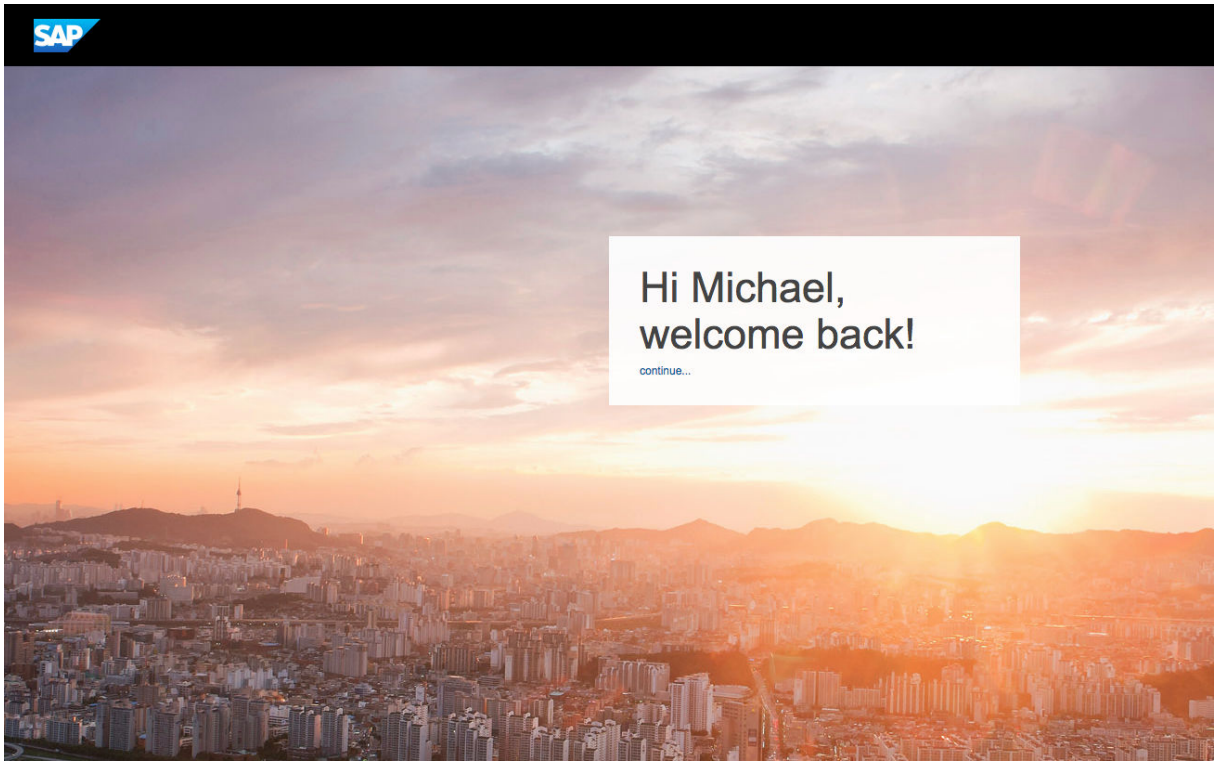
1.7.9 Case Study: Gamified Help Desk Application

In this case study, a demo application will be gamified in order to demonstrate the implementation and configuration of a gamification concept step by step.

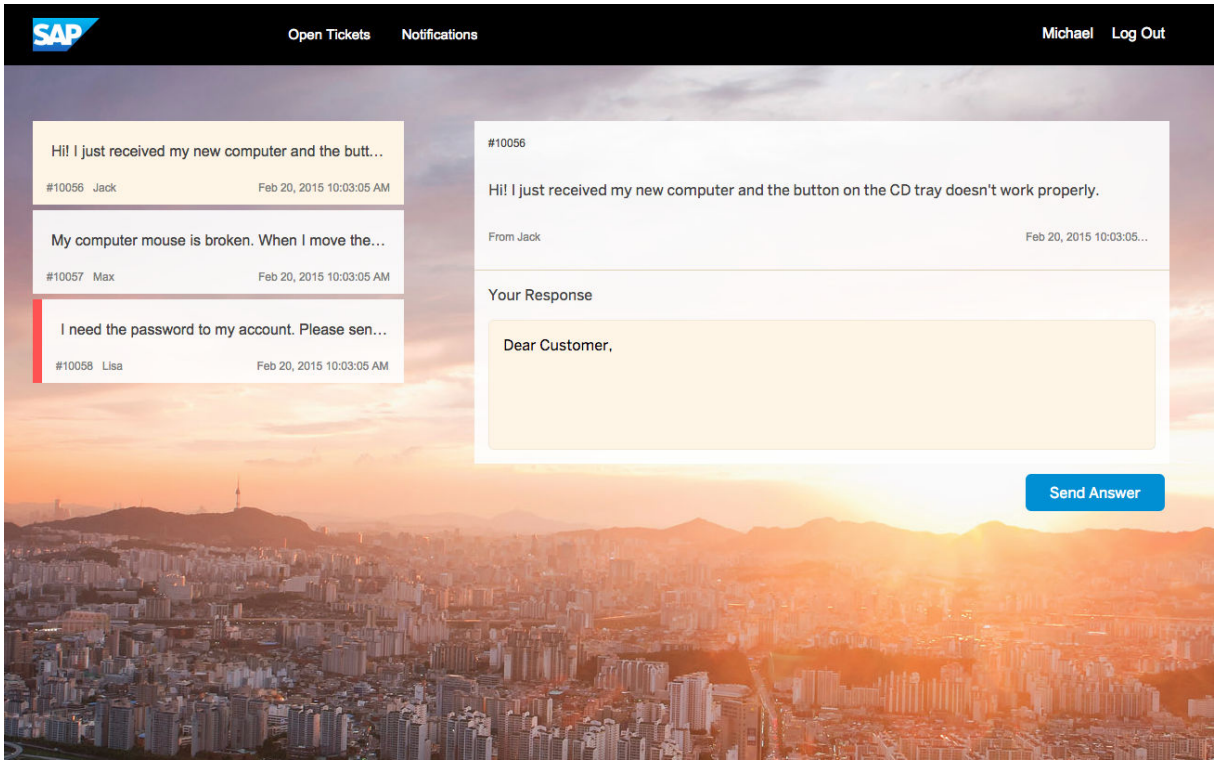
The demo host application is a “Help Desk” software, which is typically used by call center employees. Customers can create tickets (for an issue with software or hardware, for example) and call center employees can process these tickets.



The image below shows the welcome screen of the Help Desk application. The welcome screen appears once the user is successfully authenticated using the identity provided. The user must have the role `helpdesk`. The assignment of roles is described in page Roles [page 636].



Once the user is logged on, an overview of the open tickets is displayed on the left side (see the picture below). The user can select a ticket and process it by entering comments. No further actions are necessary in the demo application. Once the user has entered an appropriate response, the user can click *Send Answer* to process the ticket.



The demo application already includes a gamification implementation.

1.7.9.1 Configure Available Subscription

Context

The demo application (Help Desk) will be automatically subscribed for each subaccount that is subscribed to the gamification service.

The gamification service has already been integrated within the demo application. Events such as the processing of tickets will be sent to the gamification service of the subaccount subscription for example, and the achievements are going to be retrieved by the corresponding interfaces.

Since the gamification service and the demo applications are subscriptions, a destination has to be enabled in order to allow communication between the services. A technical user is also required in order to allow secure communication.

Procedure

The Help Desk app can be accessed via the menu *Help* > *Open Help Desk*. The following link will be used: https://< SUBSCRIPTION_URL>/helpdesk. The role `helpdesk` must be granted to the user.

1.7.9.1.1 Assign Help Desk Role

Context

The user requires the role `helpdesk` in order to access the help desk application.

Procedure

1. Go to the *Services* tab in your SAP BTP cockpit subaccount.
2. Click the *Gamification Service* tile.
3. Click on the *Configure Gamification Service* link.
4. Go to the *Roles* tab.
5. Assign the role `helpdesk` to your user.

Related Information

Managing Roles [page 1710]
Assign Gamification Roles [page 624]

1.7.9.1.2 Create Technical User

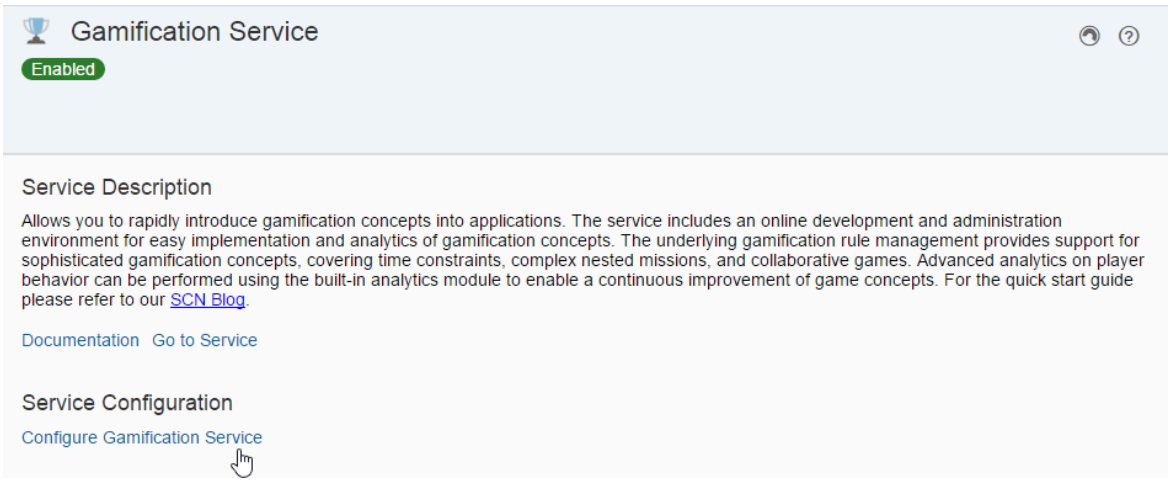
The destination requires a technical user for secure communication between your application and the gamification service subscription.

Context

> **i Note**
>
> You can request user IDs at the SAP Service Marketplace: http://service.sap.com/request-user SAP Service Marketplace users are automatically registered with the SAP ID service, which controls user access to SAP BTP.

Procedure

1. Request a technical user via SMP. (You can use your subaccount user as well, but this is not recommended for security reasons.)
2. In the SAP BTP cockpit, choose the *Services* tab.
3. Click the *Gamification Service* tile.
4. Click on the *Configure Gamification Service* link.



5. Go to the *Roles* tab.
6. Assign the `AppAdmin` role.

Related Information

Managing Roles [page 1710]

1.7.9.2 Deploy HelpDesk App

Prerequisites

You have an Eclipse IDE with SAP BTP tools.

For more information about how to install the SAP BTP tools, see *Eclipse Tools*.

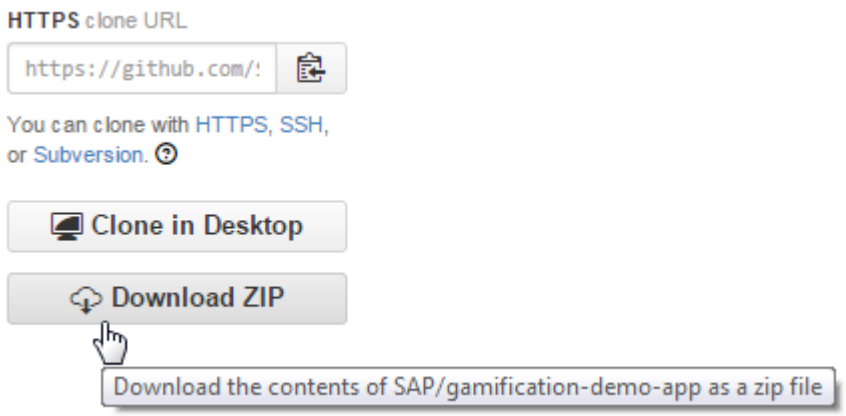
Context

The demo application's (Help Desk) source code is also available in GitHub .

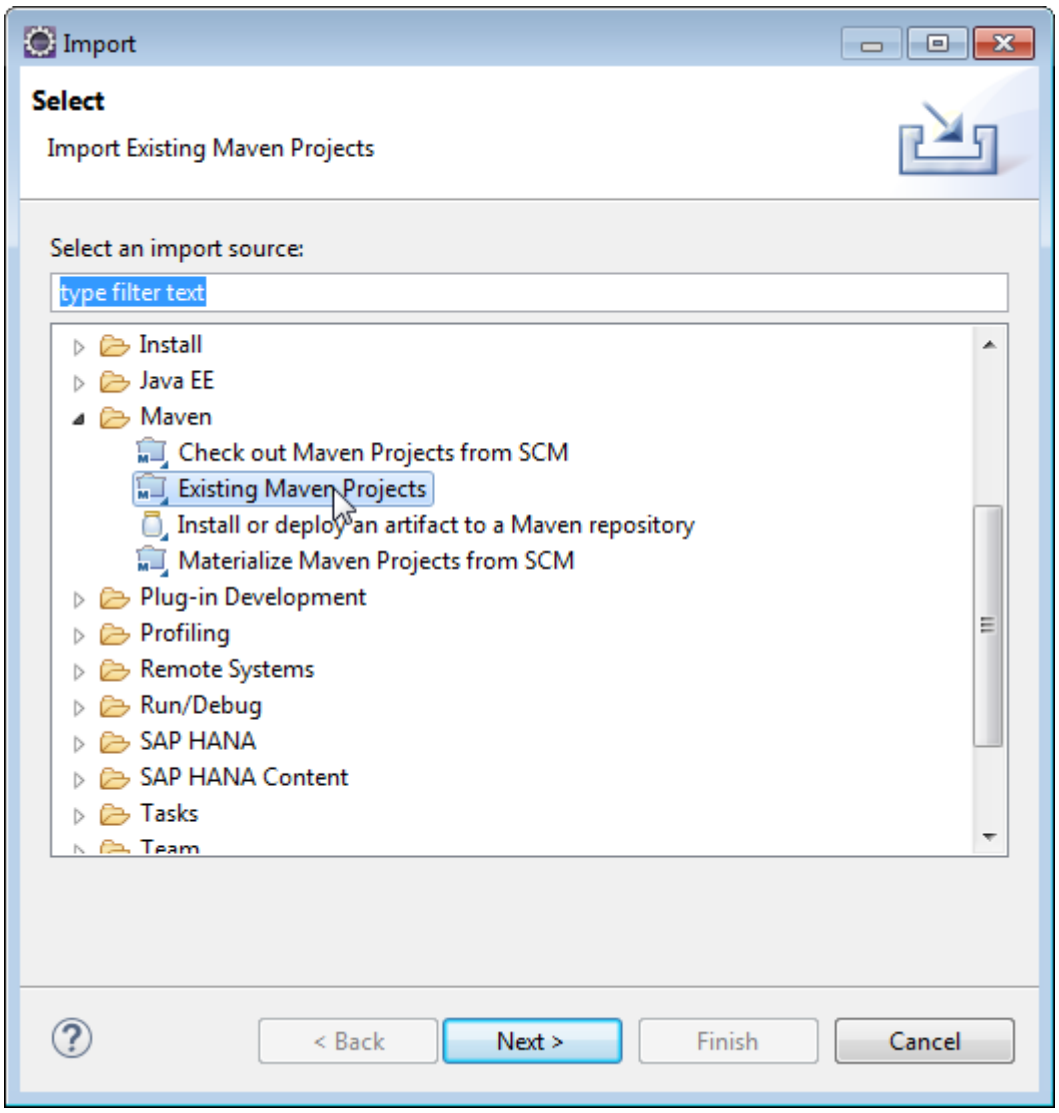
This section explains how to set up an Eclipse project, deploy the demo application on SAP BTP, and configure it to run with your gamification service subscription.

Procedure

1. Download sources as a zip from GitHub .



2. Extract the archive.
3. Open Eclipse with SAP BTP tools and choose *File* > *Import*.
4. Choose *Maven* > *Existing Maven Projects*.



5. Choose the folder containing the demo application sources and choose *Finish*.
6. Deploy and start the demo application on the cloud from Eclipse IDE. Select Java Web as a Runtime.

   For more information, see *Deploying on the Cloud from Eclipse IDE*.
7. Configure destinations and roles for the deployed application. Use the same configuration as described in section Configure Available Subscription [page 693].

1.7.9.3 Gamification Design

The host application without the application does not allow the user (call center employee) to see any feedback on his/her daily work. The user does not really know how s/he performs compared to other colleagues either.

The requirement for gamification in the demo applications is to intrinsically motivate the users with instant feedback (achievements). Collaborative feedback will be introduced, and the progress for each individual user will be visible as well as the performance compared to others.

To meet the introduced gamification requirements, an example gamification design is introduced. All users (call center employees) are considered as players where the gamification concept will apply.

Points Categories

Points are introduced to represent the experience of the users:

“Experience Points” (XP)

“Critical Tickets” (CT)

Levels

Based on the number of experience points a user gains, s/he can reach different levels. Three levels are introduced:

“Novice” - this level can be reached already with 0 “Experience Points”

“Competent” - this level can be reached once the user has gained 10 “Experience Points”

“Expert” - this level can be reached once the user has gained 50 “Experience Points”

Badges

Based on the successful completion of a mission, the user will gain a badge. The following badges are introduced:

“Troubleshooting Champion”

Missions

Missions will be introduced to motivate continuous efforts. The following missions will be introduced:

“Troubleshooting”

Rules

For each processed ticket, the user will gain 1 “Experience point”.

For each processed ticket categorized as “critical”, the user will gain 2 additional “Experience Points” to motivate him or her to solve critical tickets with higher priority.

For each processed ticket categorized as “critical”, the user will gain 1 “Critical Tickets” point.

Once a user has processed 5 critical tickets (gained 5 “Critical Tickets” points), the “Troubleshooting” mission is completed.

Once the mission troubleshooting is completed, the user will gain the “Troubleshooting Champion” badge.

1.7.9.4 Gamification Concept – Content Generation

The gamification concept introduced above can be generated automatically within the gamificationworkbench. The generated gamification concept is designed for the demo application only and is designed to provide an example of a gamification concept.

The demo content for the *Help Desk* application can be generated in the *OPERATIONS* tab. You need to have the `TenantOperator` role. Go to "Demo Content Creation" (shown in the picture below) and select the *Create HelpDesk Demo* button. After a short while you will see a notification `Gamification concept successfully created.` once the content generation was successful. The demo content has been generated into a new app: *HelpDesk*.



The generated gamification concept contains more gamification elements than described in Switch Apps [page 643] to provide additional examples.

1.7.9.5 Gamification Concept – Manual Implementation

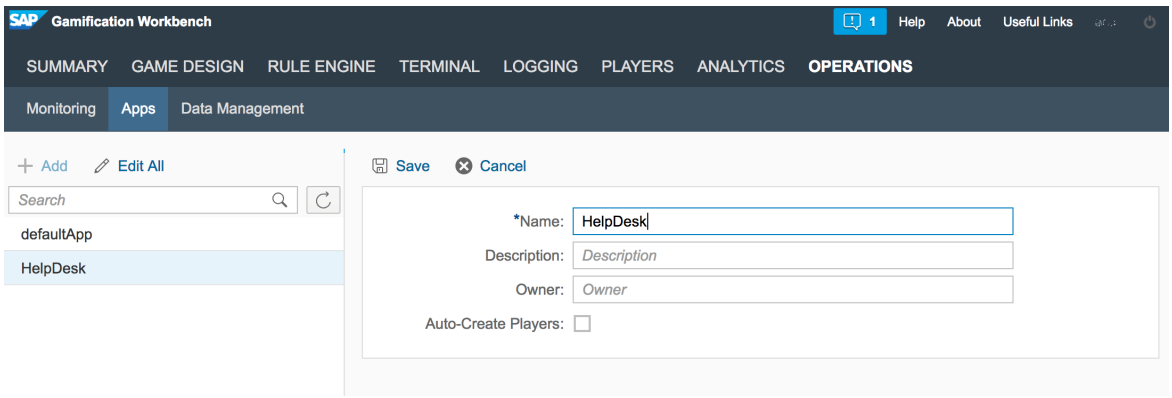
The following sections describe how the gamification design is realized in the gamification workbench.

1.7.9.5.1 Create Application

The gamification workbench makes it possible to manage gamification concepts for multiple apps. An app must be created before the gamification concept can be implemented.

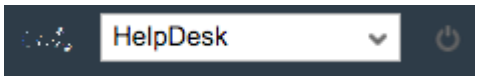
Procedure

1. Go to the *OPERATIONS* tab. The user must have the `TenantOperator` role.
2. Go to *Apps*.
3. Press the *Add* button.
4. Enter App name: “HelpDesk”.
5. Optional: Enter an app description.
6. Optional: Enter owner.
7. Click on *Save*.



Next Steps

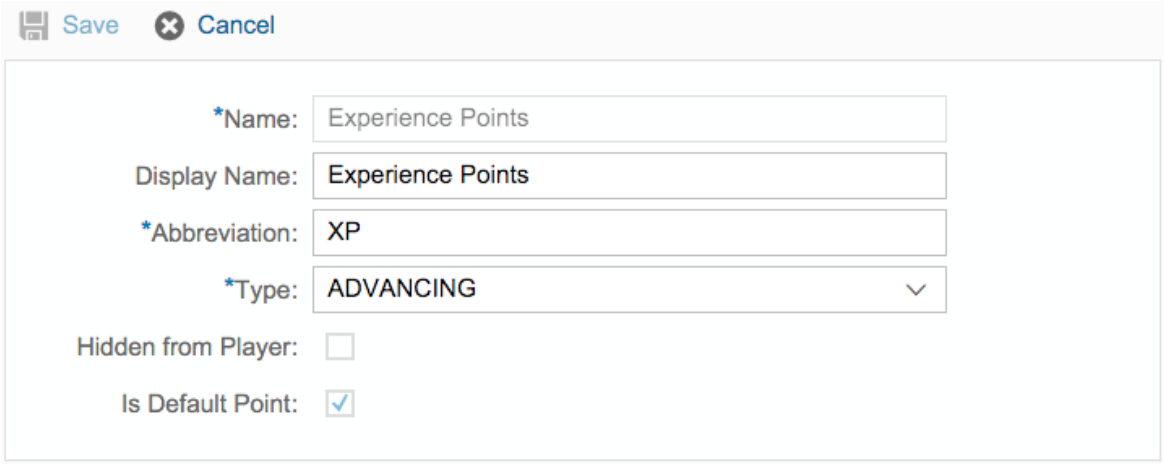
Once the app has been created, it must be selected in the top right corner so that the gamification concept can be implemented for it.



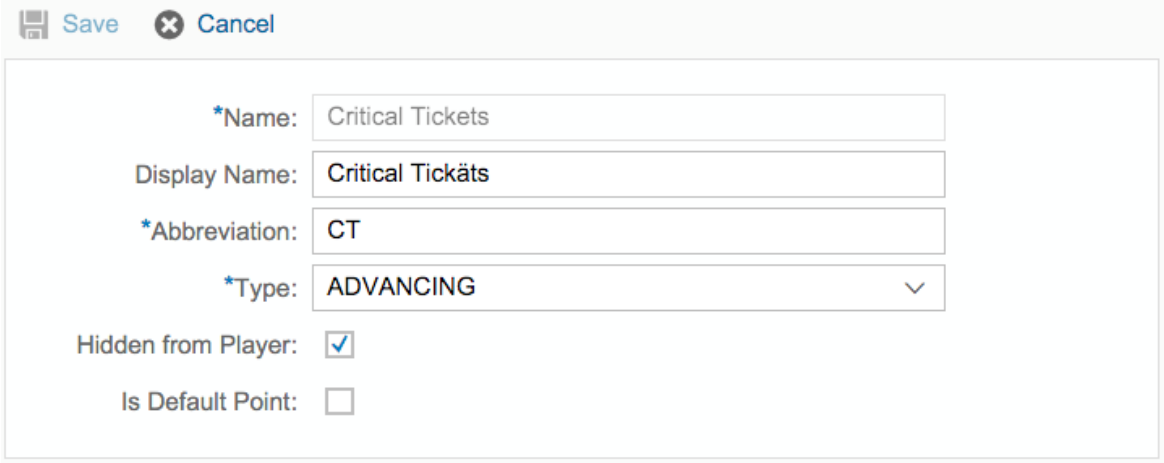
1.7.9.5.2 Create Point Categories

Procedure

1. Go to *Game Design* in the navigation menu and select *Points*.
2. Press *Add*.
3. Enter Name: “Experience Points”.
4. Enter Abbreviation: “XP”.
5. Select point type: “ADVANCING”.
6. Press *Create*.

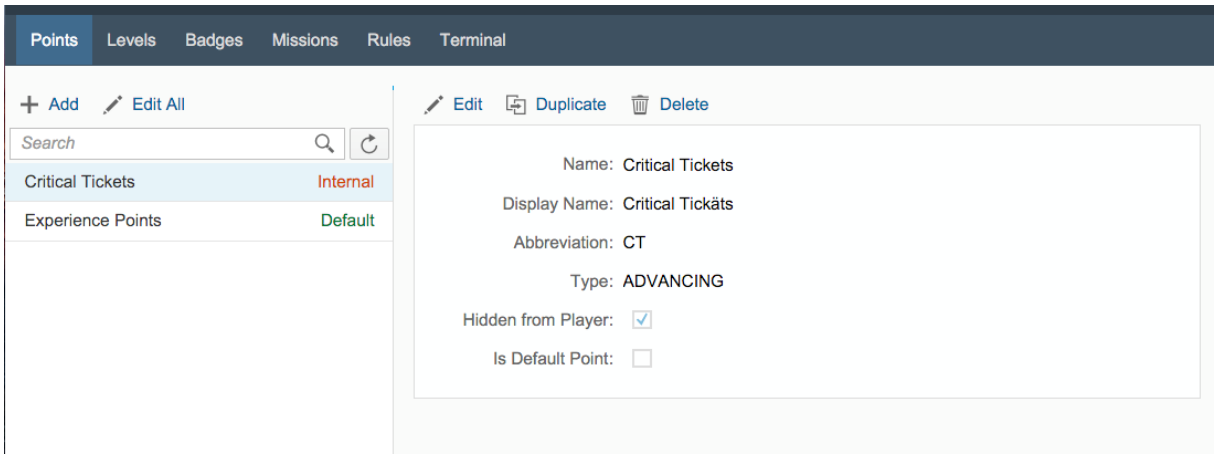


7. Press *Add*.
8. Enter Name: “Critical Tickets”.
9. Enter Abbreviation: “CT”.
10. Select point type: “ADVANCING”.
11. Check *Hidden from Player*
12. Press *Create*.



Results

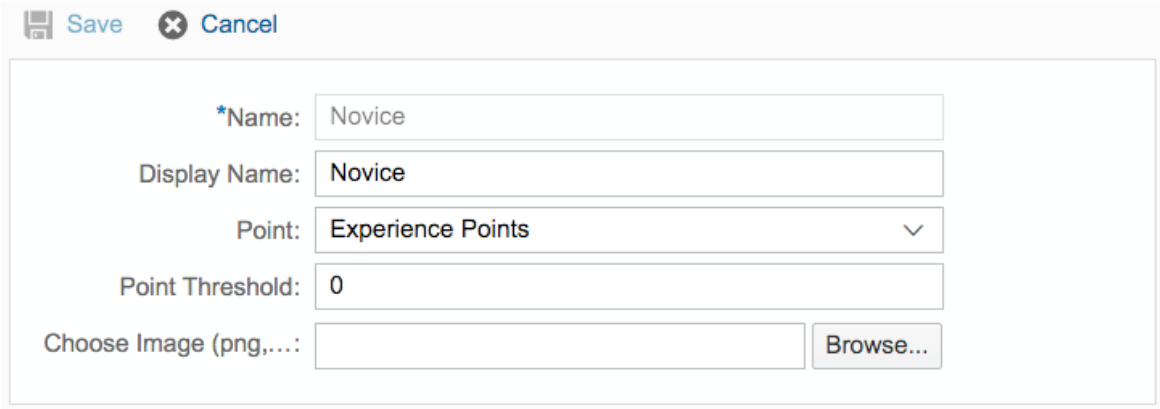
You should now see both point categories (“Experience Points” and “Critical Tickets”) in the list for Points.



1.7.9.5.3 Create Levels

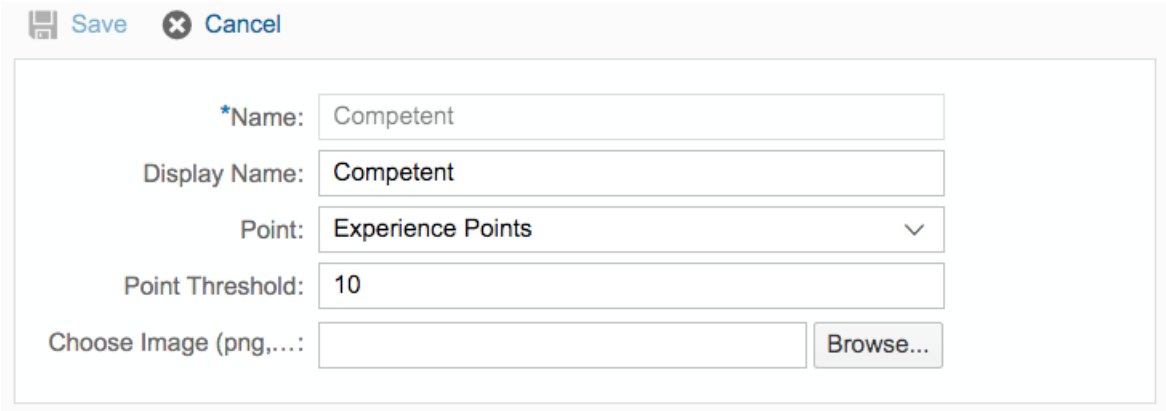
Procedure

1. Go to *Game Design* in the navigation menu and select *Levels*.
2. Press *Add*.
3. Enter Name: “Novice”.
4. Select Points: “Experience Points”.
5. Enter Threshold: “0”.
6. Press *Create*.

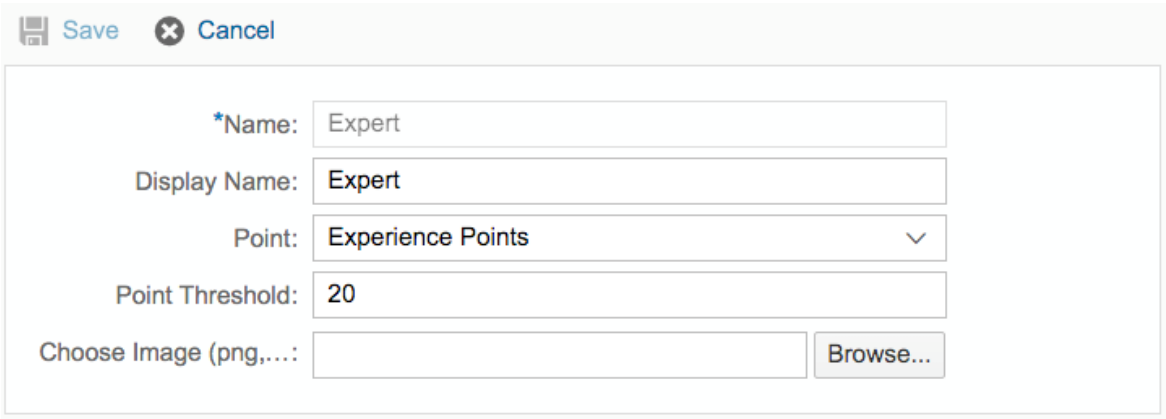


7. Press *Add*.
8. Enter Name: “Competent”.
9. Select Points: “Experience Points”.

10. Enter Threshold: “10”.
11. Press *Add*.

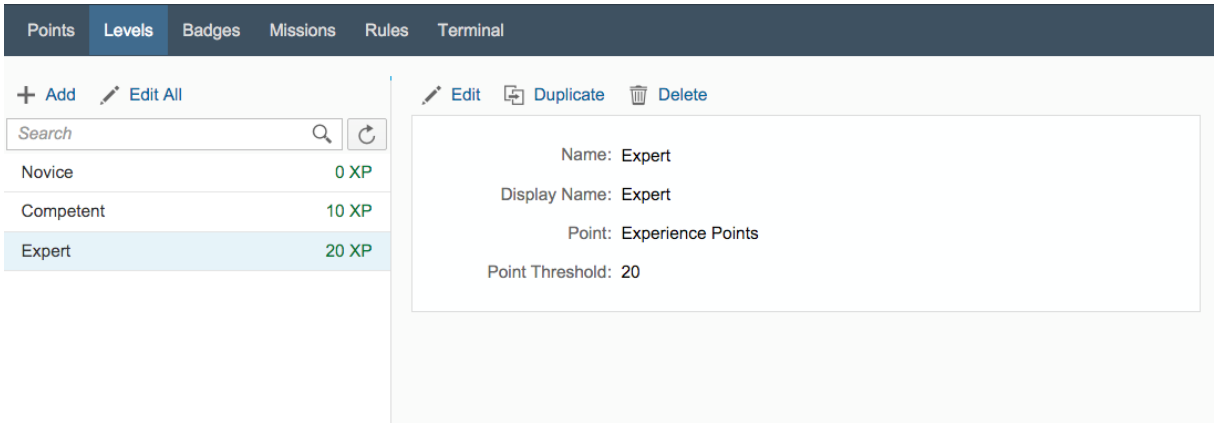


12. Press *Add*.
13. Enter Name: “Expert”.
14. Select Points: “Experience Points”.
15. Enter Threshold: “50”.
16. Press *Create*.



Results

You should now see all three levels (“Novice”, “Competent”, and “Expert”) in the list for Levels.



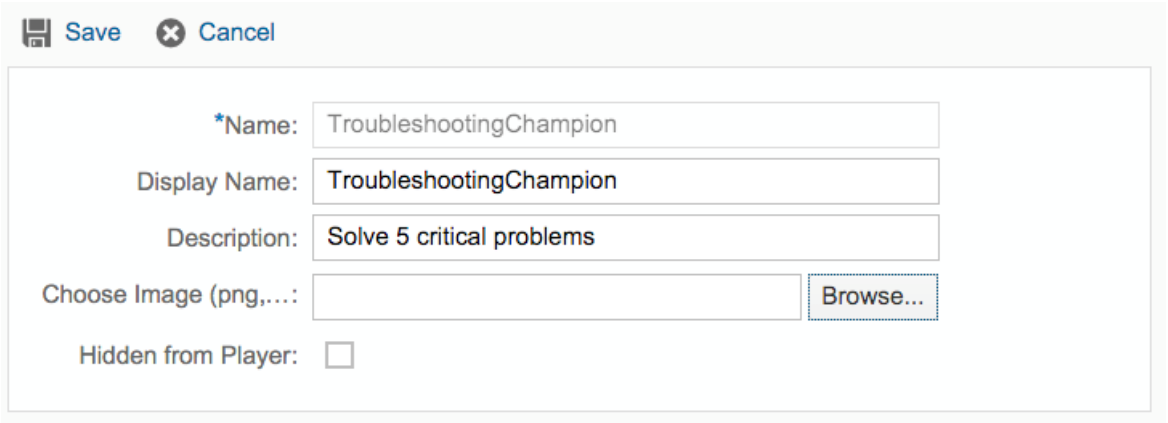
1.7.9.5.4 Create Badges

Procedure

1. Go to *Game Design* in the navigation menu and select *Badges*.
2. Press *Add*
3. Choose file for upload: “Troubleshooting_Chamption.png”.

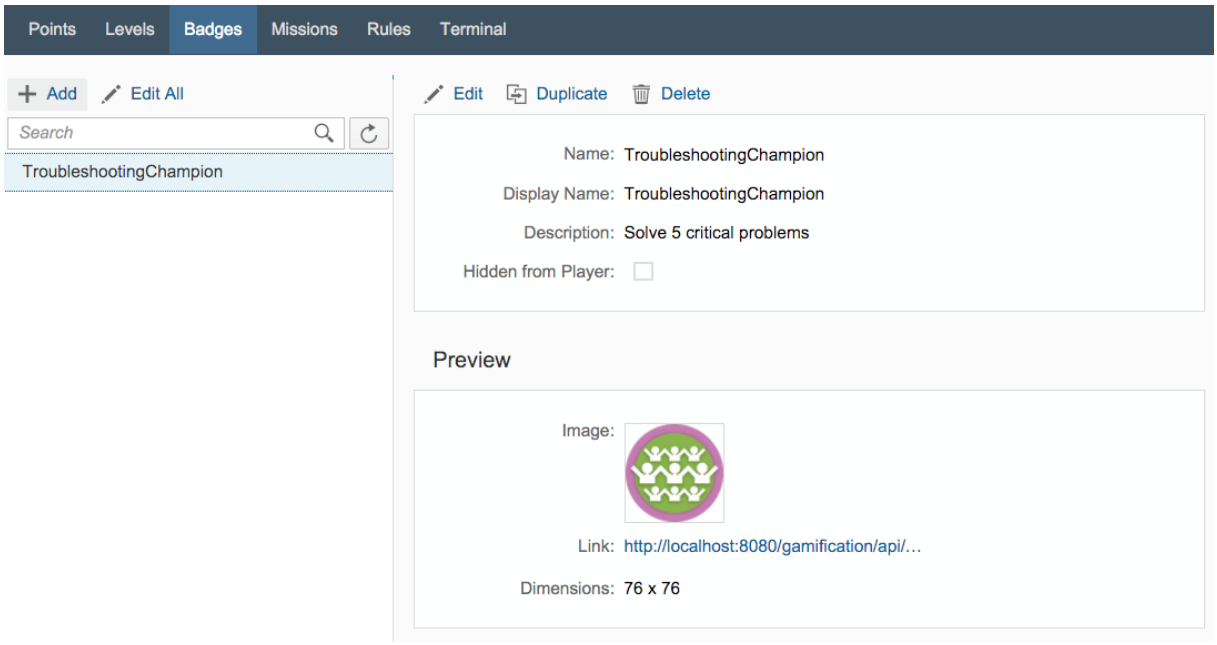


4. Enter Name: “Troubleshooting Champion”.
5. Enter Description: “Solve 5 critical problems”.
6. Press *Create*.



Results

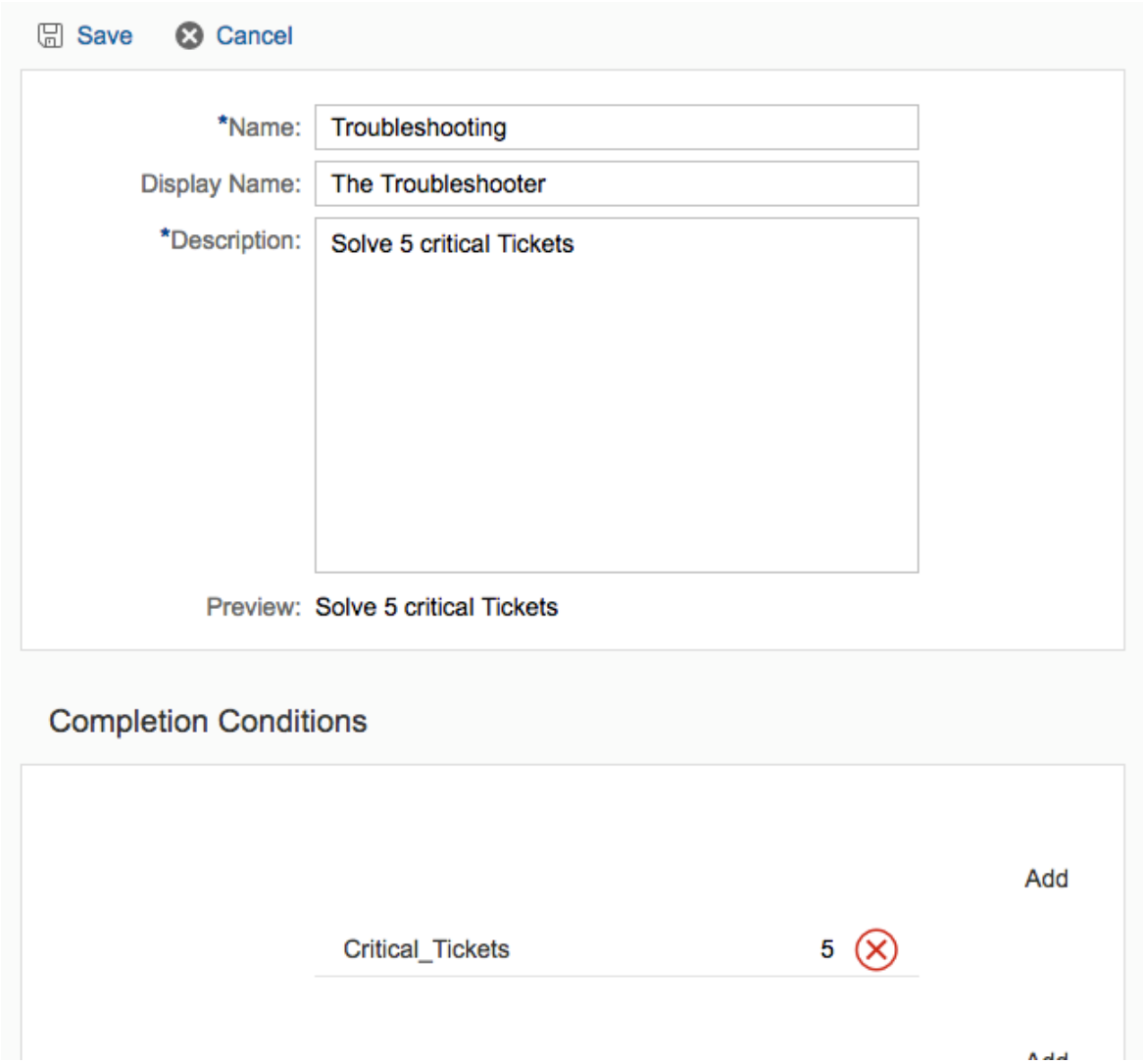
You should now see all badges (“Troubleshooting Champion”) in the list for Badges.



1.7.9.5.5 Create Missions

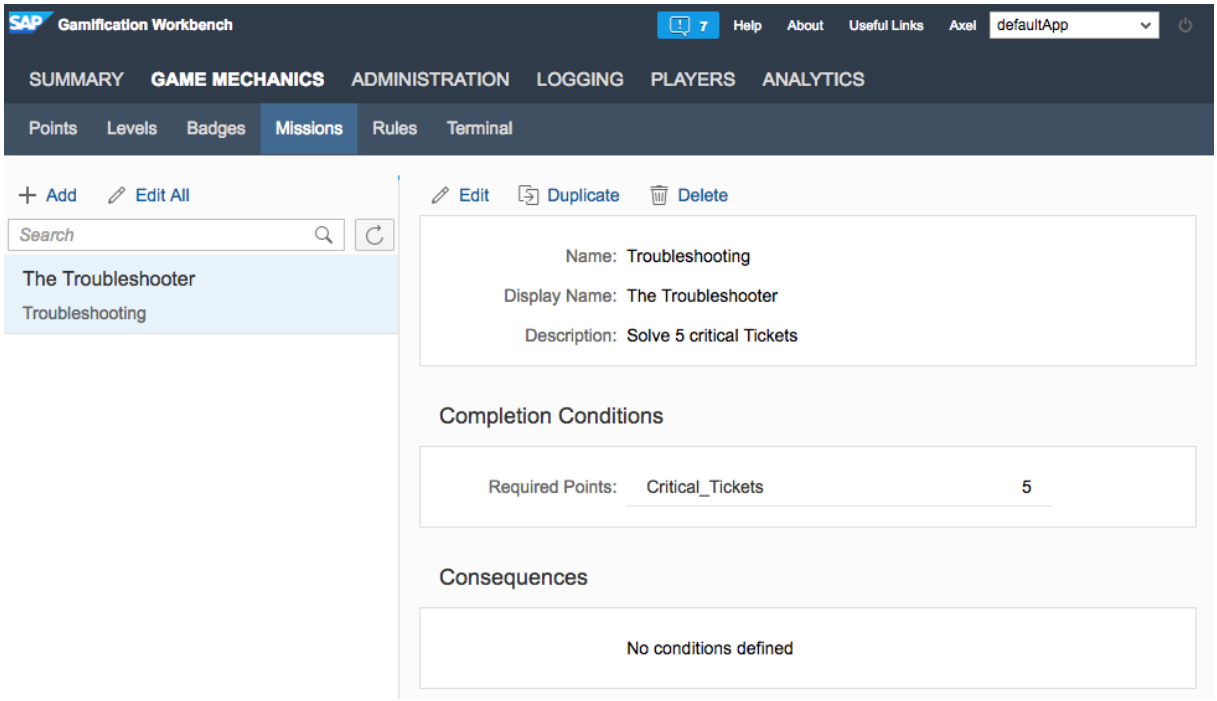
Procedure

1. Go to *Game Design* in the navigation menu and select *Missions*.
2. Press *Add*.
3. Enter Name: “Troubleshooting”.
4. Enter Description: “Solve 5 critical tickets.”.
5. Enter Consequence: “Troubleshooting Champion”.
6. Add a point condition: Press the *Add* button and select a point and threshold. Confirm the dialog with the *Add* button.
7. Press *Create*.



Results

You should now see all missions (“Troubleshooting”) in the list for Missions.



1.7.9.5.6 Creating Rules

Context

To create rules, you have to perform the following activities:

- Give Experience Points [page 707]
- Give Experience Points for a Critical Mission [page 707]
- Give Critical Ticket Points [page 708]
- Assign Troubleshooting Mission [page 708]
- Complete Troubleshooting Mission [page 708]

1.7.9.5.6.1 Give Experience Points

Procedure

1. Go to *Game Design* in the navigation menu and select *Rules*.
2. Go to *All Rules* in the sub navigation.
3. Press *Add*.
4. Enter Name: “GiveXP”
5. Enter Description: “Give Experience Points for processed ticket.”
6. Enter the following text for the trigger:

```
$event : EventObject(type=='solvedProblem', $playerid:playerid) from entry-
point eventstream
```

7. Enter the following text for the consequence:

```
updateAPI.givePoints($playerid, 'Experience Points', 1, 'Ticket processed');
update(engine.getPlayerById($playerid));
```

8. Press *Save*. The *Activate on Engine Update* checkbox must be checked.

1.7.9.5.6.2 Give Experience Points for a Critical Mission

Procedure

1. Press *Add*.
2. Enter Name: “GiveXPCritical”
3. Enter Description: “Give additional Experience Points for critical ticket.”
4. Enter the following text for the trigger:

```
$event : EventObject(type=='solvedProblem', data['relevance']=='critical',
$playerid:playerid) from entry-point eventstream
```

5. Enter the following text for the result:

```
updateAPI.givePoints($playerid, 'Experience Points', 2, 'Critical ticket
processed');
update(engine.getPlayerById($playerid));
```

6. Press *Save*. The *Activate on Engine Update* checkbox must be checked.

1.7.9.5.6.3 Give Critical Ticket Points

Procedure

1. Press *Add*.
2. Enter Name: “GiveCT”
3. Enter Description: “Give Critical Ticket Points for processed ticket.”
4. Enter the following text for the trigger:

```
$event : EventObject(type=='solvedProblem', $playerid:playerid) from entry-
point eventstream
```

5. Enter the following text for the consequence:

```
updateAPI.givePoints($playerid, 'Critical Tickets', 1, 'Critical ticket
processed');
update(engine.getPlayerById($playerid));
```

6. Press *Save*. The *Activate on Engine Update* checkbox must be checked.

1.7.9.5.6.4 Assign Troubleshooting Mission

Procedure

1. Press *Add*.
2. Enter Name: “AssignMissionTS”
3. Enter Description: “Assign Troubleshooting mission.”
4. Enter the following text for the trigger:

```
$p : Player($playerid : uid)
$event : EventObject(type=='initPlayerForApp', $playerid==playerid) from
entry-point eventstream
```

5. Enter the following text for the consequence:

```
updateAPI.addMissionToPlayer($playerid, 'Troubleshooting');
update($p);
```

6. Press *Save*. The *Activate on Engine Update* checkbox must be checked.

1.7.9.5.6.5 Complete Troubleshooting Mission

Procedure

1. Press *Add*.

2. Enter Name: “CompleteMissionTS”
3. Enter Description: “Complete Troubleshooting Mission.”
4. Enter the following text for the trigger:

```
$p : Player($playerid : uid);
eval(queryAPI.hasPlayerMission($playerid, 'Troubleshooting') == true)
eval(queryAPI.getPointsForPlayer($playerid, 'Critical Tickets').getAmount()
>= 5)
```

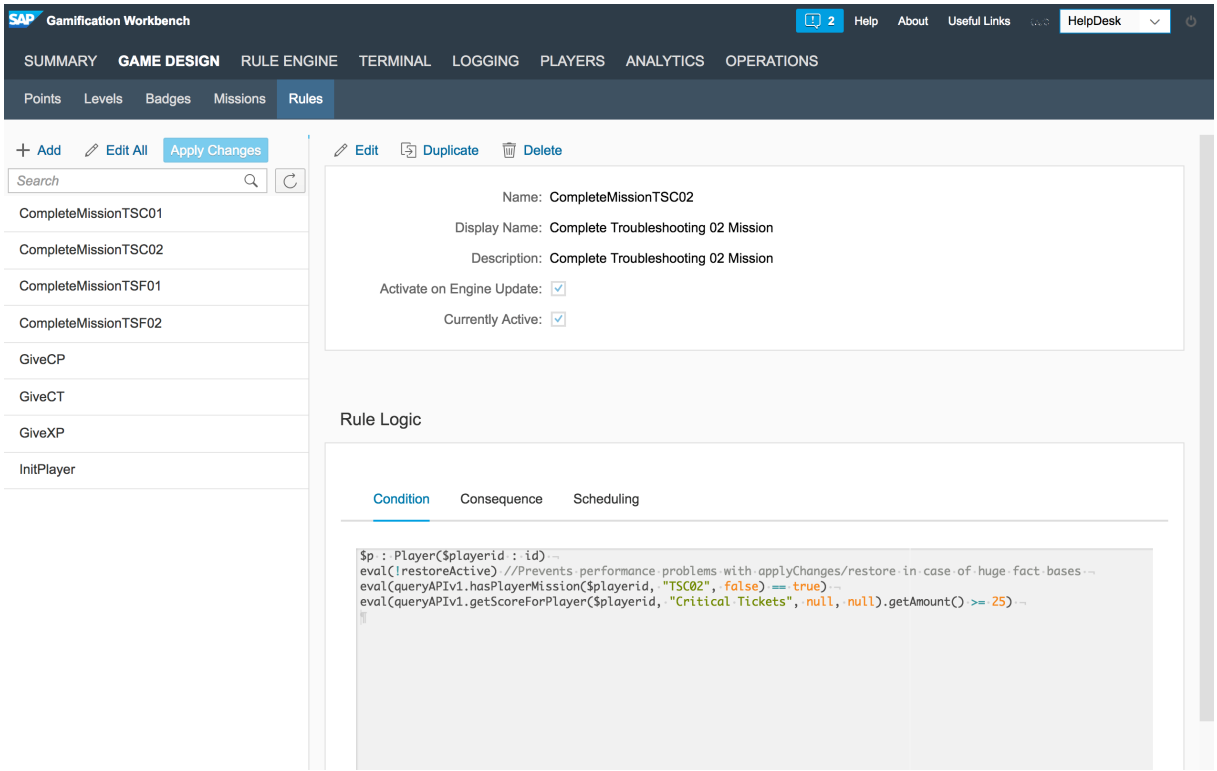
5. Enter the following text for the consequence:

```
updateAPI.completeMission($playerid, 'Troubleshooting');
updateAPI.addBadgeToPlayer($playerid, 'Troubleshooting Champion', 'You solved
5 critical tickets!');
update($p);
```

6. Press *Save*. The *Activate on Engine Update* checkbox must be checked.

1.7.9.5.6.6 Result

You should now see the created rules in the list for Rules.



Deploy and Activate Rules

7. Press Apply Changes.

8. Confirm the dialog.

Results

All rules are shown as active without any issue warnings.

1.8 SAP Git Service

Store and version source code in Git repositories.

The SAP Git service lets you store and version the application source code. It is based on Git, the widely used open-source system for revision management of source code that facilitates distributed and concurrent large-scale development workflows.

Git is a widely used open source system for revision management of source code that facilitates distributed and concurrent large-scale development workflows.

You can use any standard compliant Git client to connect to the SAP Git service. Many modern integrated development environments, including but not limited to SAP Web IDE, provide tools for working with Git. There are also native clients available for many operating systems and platforms.

Environment

This service runs in the Neo environment.

Features

Records differences between versions	Only the differences between versions are recorded allowing for a compact storage and efficient transport.
Cost-effective and simple	Create and merge branches supporting a multitude of development styles. Git is widely used and supported by many tools and is highly distributed. A clone of a repository contains the complete version history.
Operations on local repository clone	Perform almost all operations locally and thus very fast and without need to be permanently online. Only required when synchronizing with the Git service.

Restrictions

The SAP Git service is a dedicated service for source code versioning.

While Git can manage and compare text files very efficiently, it was not designed for processing large files or files with binary content, such as libraries, build artifacts, multimedia files (images or movies), or database backups. Consider using the document service or some other suitable storage service for storing such content.

To ensure best possible performance and health of the service, the following restrictions apply:

- The size of an individual file cannot exceed 20 MB. Pushes of changes that contain a file larger than 20 MB are rejected.
- The overall size of the bare repository stored in the SAP Git service cannot exceed 500 MB.
- The number of repositories per subaccount is not currently limited. However, SAP may take measures to protect the SAP Git service against misuse.

Third-Party Notice

The SAP Git service makes use of the Git-Icon-1788C image made available by Git (https://git-scm.com/downloads/logos) under the Creative Commons Attribution 3.0 Unported License (CC BY 3.0) http://creativecommons.org/licenses/by/3.0 .

Related Information

Managing Repositories [page 712]
Working with Repositories [page 716]
Security [page 720]
Best Practices [page 723]
Git
SAP Web IDE
What is Document Service [page 547]

1.8.1 Managing Repositories

In the SAP BTP cockpit, you can create and delete Git repositories, as well as lock and unlock repositories for write operations. In addition, you can monitor the current disk consumption of your repositories and perform garbage collections to clean up and compact repository content.

Related Information

Create Repositories [page 712]
Change the State of Repositories [page 714]
Delete Repositories [page 715]
Clean Repositories [page 716]

1.8.1.1 Create Repositories

In the SAP BTP cockpit, you can create Git repositories for your subaccounts.

Prerequisites

You must be an administrator of the subaccount.

Context

> **i Note**
>
> To create a repository for the static content of an HTML5 application, see Create an HTML5 Application [page 1126].

Procedure

1. Log on to the SAP BTP cockpit, and select the subaccount.
2. From the navigation area, choose *Repositories* > *Git Repositories*.
3. Choose *New Repository* and enter the following data.

Data for Repository Creation

Field	Entry
Name	(Mandatory) A unique name starting with a lowercase letter, followed by digits and lowercase letters. The name is restricted to 30 characters.
Description	(Optional) A descriptive text for the repository. You can change this description later on.
Create empty commit	An initial empty commit in the history of the repository. This might be useful if you want to import the content of another repository.

4. Choose *OK*.
5. To navigate to the details page of the repository, click its name.

Results

The URL of the Git repository appears under *Source Location* on the detail page of the repository. You can use this URL to access the repository with a standard-compliant Git client. You cannot use this URL in a browser to access the Git repository.

Related Information

Navigate in the Cockpit
Create an HTML5 Application [page 1126]
Assign Developers to a Repository [page 713]

1.8.1.1.1 Assign Developers to a Repository

Permissions for Git repositories are granted based on the subaccount member roles that are assigned to users. To grant a subaccount member access to a Git repository, assign one of these roles: *Administrator*, *Developer*, or *Support User*.

Prerequisites

- You must be an administrator of the subaccount.
- New members to be added must already have SAP user IDs.

Context

For details about the permissions associated with the individual roles, see Security [page 720].

Procedure

1. Log on to the SAP BTP cockpit.
2. Assign members to subaccounts and define their roles.

   Make sure that you assign at least one of these roles: Administrator, Developer, or Support User.

Related Information

Managing Member Authorizations in the Neo Environment [page 1301]
User and Member Management

1.8.1.2 Change the State of Repositories

In the SAP BTP cockpit, you can change the state of a Git repository temporarily to READ ONLY to block all write operations.

Prerequisites

You must be an administrator of the subaccount.

Procedure

1. Log on to the SAP BTP cockpit, and select the required subaccount.
2. In the list of Git repositories, locate the repository you want to work with and follow the link on the repository's name.
3. On the details page of the repository, choose *Set Read Only*.

Results

The state flag of the repository changes from *ACTIVE* to *READ ONLY* and all further write operations on this repository are prohibited.

> **i** Note
>
> To unlock the repository again and allow write access, choose *Set Active* on the details page of the repository.

1.8.1.3 Delete Repositories

In the SAP BTP cockpit, you can delete a Git repository unless it is associated with an HTML5 application. In this case, delete the HTML5 application.

Prerequisites

You must be an administrator of the subaccount.

Context

> **⚠** Caution
>
> Be very careful when using this command. Deleting a Git repository also permanently deletes all data and the complete history. Clone the repository to some other storage before deleting it from the SAP Git service in case you need to restore its content later on.
>
> For more information, see Clone Repositories [page 717].

Procedure

1. Log on to the SAP BTP cockpit, and select the appropriate subaccount.
2. From the navigation area, choose *Repositories* > *Git Repositories*.
3. Select the repository you want to delete.
4. From the *Actions* column, choose the delete icon ( ).
5. Confirm that you want to delete the repository.

1.8.1.4 Clean Repositories

In the SAP BTP cockpit, you can trigger a garbage collection for a repository to clean up unnecessary objects and compact the repository content aggressively.

Prerequisites

You are an administrator of the subaccount.

Context

Perform this operation from time to time to ensure the best possible performance for all Git operations. The SAP Git service also automatically runs normal garbage collections periodically.

> **i** Note
>
> This operation might take a considerable amount of time and may also impact the performance of some Git operations while it is running.

Procedure

1. Log on to the SAP BTP cockpit, and select the subaccount.
2. From the navigation area, choose *Repositories* > *Git Repositories*.
3. Select the repository you want to work with and click the repository's name.
4. On the details page of the repository, choose *Garbage Collection*.

Results

The garbage collection runs in the background. You can use the Git repository without restrictions while the process is running.

1.8.2 Working with Repositories

Git supports many commands for working with repositories.

We assume that you are familiar with Git concepts, and that you have access to a suitable Git client, for example, SAP Web IDE for performing Git operations.

If you are new to Git, we strongly recommend that you read a text book about it, and consult the Best Practices guide before using the service.

Related Information

Using Source Control (Git) in SAP Web IDE
Best Practices [page 723]

1.8.2.1 Determine the Repository URL

The URL of the Git repository is shown under *Source Location* on the details page of the repository. Use this URL to access the repository using a Git client.

Prerequisites

In the subaccount where the repository resides, you must be a subaccount member who is assigned the role *Administrator*, *Developer*, or *Support User*.

Procedure

1. Log on to the SAP BTP cockpit, and select the required subaccount.
2. From the navigation area, choose *Repositories* > *Git Repositories*.
3. Select the repository you want to work with and follow the link on the repository's name.
4. In the *Source Location* panel, copy the link labeled *Git Repository URL*.

1.8.2.2 Clone Repositories

You need to clone the Git repository of your application to your development environment.

Procedure

1. In the cockpit, copy the link to the Git repository of your application.
   a. Log on with a user who is a subaccount member to the SAP BTP cockpit.

b. From the navigation area, choose *Applications* > *HTML5 Applications*.
c. Click your newly created application.
d. Switch to the *Versioning* tab.
e. Under *Source Location*, copy the link that points to the Git repository of your application.

2. You can use any Git client to clone a repository from the SAP Git service. To use the Git command line client, execute these steps:
   a. Enter the following command:

      ```
      $ git clone <repository URL>.
      ```

   b. If prompted, enter your SCN user ID and password.

1.8.2.3 Fetch Changes from Repositories

The Git `fetch` operation transfers changes from the remote repository to your local repository.

Prerequisites

- You must be a subaccount member who is assigned the role *Administrator*, *Developer*, or *Support User*.
- You have cloned the repository to your workspace, see Clone Repositories [page 717].

Context

Refer to the SAP Web IDE documentation if you want to fetch changes to SAP Web IDE. Otherwise, see the documentation of your Git client to learn how to fetch changes from a remote Git repository.

Procedure

1. Execute a `fetch` or a `pull` command with your Git client.
2. Authenticate yourself with your SAP ID credentials.

Related Information

SAP Web IDE

1.8.2.4 Push Changes to Repositories

The Git `push` operation transfers changes from your local repository to a remote repository.

Prerequisites

- You must be a subaccount member who is assigned the role *Administrator* or *Developer*.
- You have already committed the changes you want to push in your local repository.
- You have ensured that the e-mail address in the push commit matches the e-mail address you registered with the SAP ID service.

Context

Refer to the SAP Web IDE documentation if you want to push changes from SAP Web IDE. Otherwise, see the documentation of your Git client to learn how to push changes to a remote Git repository.

Procedure

1. Execute a `push` command with your Git client.
2. Authenticate yourself with your SAP ID credentials.

Related Information

SAP Web IDE

1.8.2.5 Browse the Content of Repositories

The SAP Git service offers a web-based repository browser that allows you to inspect the content of a repository.

Prerequisites

In the subaccount where the repository resides, you must be a subaccount member who is assigned the role *Administrator*, *Developer*, or *Support User*.

Context

The repository browser gives read-only access to the full history of a Git repository, including its branches and tags as well as the content of the files. Moreover, it allows you to download specific versions as ZIP files.

The repository browser automatically renders `*.md` Markdown files into HTML to make it easier to create documentation.

Procedure

1. Log on to the SAP BTP cockpit, and select the required subaccount.
2. From the navigation area, choose *Repositories* > *Git Repositories*.
3. Select the repository you want to work with and follow the link on the repository's name.
4. In the *Source Location* panel, click the *Repository Browser* link.

1.8.2.6 Find Commits of a Given User

You can find the commits of a given user containing the user's name and e-mail address.

Procedure

1. Clone the Git repositories of the account to which the user had write access.

   For more information, see Determine the Repository URL [page 717] and Clone Repositories [page 717].
2. On each of the Git repositories, execute the following commands:

```
$ git log --author <email address of the user>
```

```
$ git log --committer <email address of the user>
```

1.8.3 Security

Access to the SAP Git service is protected by SAP BTP roles and granted only to subaccount members.

Restrictions

You can't host public repositories or repositories with anonymous access on the SAP Git service.

Authentication

Authentication for the SAP Git service is performed against the configured platform identity provider. The following providers are supported:

- SAP ID Service
  Users can use their SAP ID Service credentials to authenticate and access the SAP Git service.
- Custom Identity Authentication tenant
  The SAP Git service supports basic authentication against a custom Identity Authentication tenant that is configured as a platform identity provider. If a subaccount is configured to use a custom Identity Authentication tenant as the platform identity provider as described in Platform Identity Provider [page 1746], then basic authentication is done against that custom Identity Authentication tenant.
  You can add members to a subaccount from the SAP ID Service as well. However, these users can't be used for basic authentication. This is because the security service doesn't support mixed use of custom platform Identity Authentication tenant users and SAP ID service users for basic authentication in the same subaccount. For more information, see the notes on basic authentication in Authentication [page 1678].
  If a custom Identity Authentication tenant is configured as the platform identity provider to grant Git permissions to users in this tenant, assign the respective roles to the respective user IDs (Pxxxxxx).

For more information on platform scopes, see Platform Scopes [page 1307].

If you've configured your Identity Authentication tenant to act as a proxy to the corporate identity provider by following the steps described in Configure Trust with Corporate Identity Provider and Choose a Corporate Identity Provider as Default, make sure that SAP BTP receives the SAML 2.0 attributes with exactly the following names:

- `first_name`: First name of the user
- `last_name`: Last name of the user
- `mail`: E-mail of the user

To achieve this, configure your corporate identity provider to send SAML 2.0 attributes with exactly the same names as mentioned above in the SAML 2.0 assertion. For more information, see Configure the User Attributes Sent to the Application.

> **i Note**
>
> Before the SAP Git service supported custom Identity Authentication tenants (until October 25, 2018), only users of the SAP ID Service could perform basic authentication when executing Git commands.

Permissions

The permitted operations depend on the subaccount member role of the user.

Read access is granted to all users who are assigned the *Administrator*, *Developer*, or *Support User* role. These users are allowed to do the following:

- Clone a repository
- Fetch commits and tags

Write access is granted to all users who are assigned the *Administrator* or *Developer* role. These users are allowed to do the following:

- Create repositories.
- Push commits
- Push tags

> **i Note**
>
> If the repository is associated with an HTML5 application, pushing a tag defines a new version for the HTML5 application. The version name is the same as the tag name.

- Create new remote branches.
- Push commits authored by other users (forge author identity).

Only users who are assigned the *Administrator* role are allowed to do the following:

- Delete repositories
- Run garbage collection on repositories
- Lock and unlock repositories
- Delete remote branches
- Delete tags
- Push commits committed by other users (forge committer identity)
- Forcefully push commits, for example to rewrite the history of a Git repository
- Forcefully push tags, for example to move the version of an HTML5 application to a different commit

For more information, see .

You can also use custom roles to grant permissions for using the SAP Git service. For more information on custom roles, see Manage Custom Platform Roles [page 1306].

Related Information

Managing Member Authorizations in the Neo Environment [page 1301]
Create a Version [page 1127]

1.8.3.1 Data Protection and Privacy

Governments place legal requirements on industry to protect data and privacy. We provide features and functions to help you meet these requirements.

> **i Note**
>
> SAP does not provide legal advice in any form. SAP software supports data protection compliance by providing security features and data protection-relevant functions, such as blocking and deletion of personal data. In many cases, compliance with applicable data protection and privacy laws is not covered by a product feature. Furthermore, this information should not be taken as advice or a recommendation regarding additional features that would be required in specific IT environments. Decisions related to data

> protection must be made on a case-by-case basis, taking into consideration the given system landscape and the applicable legal requirements. Definitions and other terms used in this documentation are not taken from a specific legal source.

Handle personal data with care. You as the data controller are legally responsible when processing personal data.

If you need to know which repositories contain Git commits of a given user that contain the user's name and e-mail address, see Find Commits of a Given User [page 720].

If you need help with this, open a ticket on BC-NEO-GIT as described in 1888290. Please indicate the user's e-mail address and the account where Git repositories reside to which this user had write access.

If you need to anonymize a user's e-mail address and name from a given Git repository this requires rewriting the history of the Git repository. This will change the IDs of all affected commits and their successor commits. For more information,

If you intend to delete a subaccount or terminate your contract you can export the Git repositories by cloning them. For more information, see Determine the Repository URL [page 717] and Clone Repositories [page 717].

Related Information

Data Protection and Privacy [page 1848]

1.8.4 Best Practices

Following best practices can help you get started with Git and to avoid common pitfalls.

If you are new to Git, we strongly recommend that you read a text book about Git, search the Internet for documentation and guides, or get in touch with the large worldwide community of developers working with Git.

- Before creating a new branch, perform a fetch or pull command.
  In general, base your work on the most recent changes of your co-workers. Otherwise, you might have to resolve conflicts before you can push changes.
- Create a new local branch for every new feature or bug-fix you want to start.
- Derive new local branches from `origin/master`
- Write meaningful commit messages.
  A commit message should describe **why** you are doing a particular change rather than describing **what** you did. The **why** helps your co-workers understand the intention and consequences of your change. The **what** on the other side can easily be deduced from the source code, that is, from the difference between the old and new version.
- Commit early and often.
  While developing larger features, create periodic checkpoints in form of commits, for example, when you are experimenting with different possible solutions. If you do something wrong, you can easily go back to the last known good commit and start over. Before pushing the feature, squeeze the checkpoints into one final commit.
- Create small commits that are easy to digest.

A commit that touches thousands of lines of code is likely to be much more difficult to understand and integrate than a commit that changes only a tiny piece of code. For example, it is usually considered bad practice to refactor your code and implement a new feature in one and the same commit.

- Avoid pushing patches that break your code.
  Git supports amendments to commits that have not yet been pushed to the SAP Git service. This means you can incrementally correct and improve your changes before you push them.
- Never rewrite a commit that has already been pushed to the SAP Git service.
  Your co-workers might already have fetched that commit to their local repositories and based their work on it. Instead, push a new commit.

> **i** Note
>
> The only valid exception to this guideline is if you accidentally pushed a secret, for example, a password, to the SAP Git service.

- Don't create dependencies on changes that have not yet been pushed.
  While Git provides some powerful mechanisms for handling chains of commits, for example, interactive rebasing, these are usually considered to be for experienced users only.
- Do not push binary files.
  Git efficiently calculates differences in text files, but not in binary files. Pushing binary files bloats your repository size and affects performance, for example, in clone operations.
- Store source code, not generated files and build artifacts.
  Keep build artifacts in a separate artifact repository because they tend to change frequently and bloat your commit history. Furthermore, build artifacts are usually stored in some sort of binary or archive format that Git cannot handle efficiently.
- Periodically run garbage collection.
  Trigger a garbage collection in the SAP BTP cockpit from time to time to compact and clean up your repository. Also run garbage collection regularly for repositories cloned to your workplace. This will minimize the disk usage and improve the performance of common Git commands.

1.9 What Is SAP Monitoring Service?

Learn more about SAP Monitoring service for SAP Business Technology Platform (SAP BTP). Monitor applications and databases.

The SAP Monitoring service for SAP BTP allows you to access application monitoring data and get notified of subscribed events. Configure custom metrics, thresholds, and alerts. Use the SAP BTP cockpit, the console client, or a REST API to manage monitoring data.

Environment

This service runs in the Neo environment.

Features

Fetch metrics of a Java application	Use the SAP BTP cockpit or the Metrics REST API to get the status of or the metrics from a Java application and its processes.
Fetch XS or HTML5 application metrics	Use the SAP BTP cockpit or the Metrics REST API to get the status of or the metrics from a HANA XS or HTML5 application.
Fetch metrics of a database system	Use the SAP BTP cockpit or the Metrics REST API to get the metrics of a selected database system to get information about its health state.
View history of metrics	Use the SAP BTP cockpit to see the history of metrics for a Java, HTML5, or HANA XS application, or for a database system.
Register availability checks	Use the SAP BTP cockpit, the console client, or the Checks REST API to retrieve or configure availability checks for Java or SAP HANA XS applications.
Define alert recipients	Use the console client to set e-mail alert notifications for an application or for all applications and database systems in a subaccount.
Receive alerts	Receive alert e-mail notifications when an application or a database system is down or responds slowly.
Configure JMX-based checks	Use the SAP BTP cockpit, the console client, or the Checks REST API to retrieve or configure JMX checks for Java applications.
Perform JMX operations	Use the SAP BTP cockpit to execute operations on JMX MBeans to monitor and manage the performance of the JVM and your Java applications.
Register custom checks	Use the SAP BTP cockpit or the Checks REST API to retrieve or configure custom checks for an HTML5 or SAP HANA XS application.

1.10 What's New for SAP Monitoring Service

2021

Component	Capability	Environment	Title	Description	Type	Available as of
Monitoring Service	Extension Suite - Development Efficiency	Neo	Alerting Channels REST API	You can now use the Alerting Channels REST API version 2 to configure a channel through which alert notifications are provided. The supported channel is a webhook or an email. See Alerting Channels API.	New	2021-09-01
Monitoring Service	Extension Suite - Development Efficiency	Neo	Configuring Custom Checks on Subaccount Level from the Cockpit	You can now configure custom checks for Java applications on subaccount level from the SAP BTP cockpit. See Availability Checks [page 732] and JMX Checks [page 738].	New	2021-06-03
Monitoring Service	Extension Suite - Development Efficiency	Neo	Configuring JMX Checks from the Cockpit	You can now configure JMX checks from the SAP BTP cockpit in addition to the Neo console client and the Checks REST API. See Configure JMX Checks for Java Applications from the Cockpit [page 739].	New	2021-03-25
Monitoring Service	Extension Suite - Development Efficiency	Neo	Checks REST API	A new REST API is available for configuring custom checks for Java, SAP HANA XS, and HTML5 applications. See Checks API.	New	2021-03-11

Related Information

2020 What's New for SAP Monitoring Service (Archive) [page 727]
2019 What's New for SAP Monitoring Service (Archive) [page 727]

2018 What's New for SAP Monitoring Service (Archive) [page 728]

1.10.1 2020 What's New for SAP Monitoring Service (Archive)

2020

Component	Capability	Environment	Title	Description	Type	Available as of
Monitoring Service	Extension Suite - Development Efficiency	Neo	Alerting	Email alert recipients are now required to confirm their email address. See set-alert-recipients [page 1538].	Changed	2020-07-02
Monitoring Service	Extension Suite - Development Efficiency	Neo	Alerting	Alerts for metrics of Java applications and for Java applications via alert webhooks now contain the process ID. See set-alert-recipients [page 1538] and Alert Webhook [page 780].	Changed	2020-05-07

Related Information

2019 What's New for SAP Monitoring Service (Archive) [page 727]
2018 What's New for SAP Monitoring Service (Archive) [page 728]

1.10.2 2019 What's New for SAP Monitoring Service (Archive)

2019

Component	Capability	Environment	Title	Description	Type	Available as of
Monitoring Service	Extension Suite - Development Efficiency	Neo	Monitoring HTML5 Applications	You can now retrieve metrics and receive alerts for HTML5 applications in the Neo environment. See Monitoring HTML5 Applications [page 753].	New	2019-01-17

Related Information

2018 What's New for SAP Monitoring Service (Archive) [page 728]

1.10.3 2018 What's New for SAP Monitoring Service (Archive)

2018

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Monitoring Service	DevOps	Neo	Alerting Channels REST API	You can now use the Alerting Channels REST API to configure a channel for receiving alert notifications on a specified URL. See Alert Webhooks [page 780].	New	2018-08-16
Monitoring Service	DevOps	Neo	REST API	A new version of the Metrics REST API is available with the following base URI: `https://api.{host}/monitoring/v2`. The new version is protected with OAuth 2.0 client credentials. See Metrics REST API for Java Applications [page 746].	New	2018-03-01

Related Information

What's New for SAP Monitoring Service [page 725]

1.11 Initial Setup

These are the preliminary requirements before you can use the service.

Prerequisites

You have set up your global account and subaccount. For an overview of the required steps, see Getting Started, Neo Environment [page 829].

Enabling the Service

The service is enabled by default. For more information, see Using Services in the Neo Environment [page 1152].

1.12 Monitoring Java Applications

To monitor whether your deployed application is up and running, register an availability check and JMX checks for it and configure alerting if the application goes down. For the alerting configuration, you use the console client or a REST API. You can also use a REST API to get the status or the metrics of a Java application and its processes.

View Monitoring Metrics of a Java Application [page 730]
: In the cockpit, you can view the custom checks created per a subaccount or the current metrics of a selected process to check the runtime behavior of your application. You can also view the metrics history of an application or a process to examine the performance trends of your application over different intervals of time or investigate the reasons that have led to problems with it. Furthermore, you can configure alerting to receive alerts for any changes to the states of these metrics.

Availability Checks [page 732]
: Create an availability check for a Java or an SAP HANA XS application to track if the application is available and to receive alerts for it.

JMX Checks [page 738]
: Registering JMX checks allows alerting on any metric that is based on JMX MBean attribute.

Generate Heap and Thread Dumps [page 744]
: Use the JMX console to generate heap and thread dumps in order to analyze the performance of a Java process.

Metrics REST API for Java Applications [page 746]
: Use this REST API to get metrics for Java applications in the Neo environment.

Checks REST API for Java Applications [page 751]

Use the Checks API to retrieve, set, update, or delete custom checks for Java applications in the Neo environment.

Related Information

Alerting [page 779]
Monitoring Database Systems [page 760]

1.12.1 View Monitoring Metrics of a Java Application

In the cockpit, you can view the custom checks created per a subaccount or the current metrics of a selected process to check the runtime behavior of your application. You can also view the metrics history of an application or a process to examine the performance trends of your application over different intervals of time or investigate the reasons that have led to problems with it. Furthermore, you can configure alerting to receive alerts for any changes to the states of these metrics. For how to set alerts, see the Related Information section. Moreover, the alert for a default metric also includes the process ID.

Prerequisites

- A Java application is deployed and started in your subaccount.
- The *readMonitoringData* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Context

Default Metrics of a Java Application
All Java applications include these default metrics. Custom metrics can also be added.

Metric	Value
Used Disc Space	What percent of the whole disc space is currently used
Requests per Minute	The number of HTTP requests processed by the Java application for the last minute
CPU Load	What percent of the CPU is used on average over the last one minute

Metric	Value
Disk I/O	How many bytes per second are currently read or written to the disc
OS Memory Usage	What percent of the operating system memory is currently used
Heap Memory Usage	What percent of the heap memory is currently used
Average Response Time	The average response time in milliseconds of all requests processed for the last minute
Busy Threads	The current number of threads that are processing HTTP requests

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. To view the custom checks for the subaccount, open *Monitoring* > *Java Custom Checks*.

   This view shows the custom checks only on subaccount level that are used for all the applications in the subaccount.
3. To view the current metrics for a process, open *Applications* > *Java Applications* in the navigation area for the subaccount.
4. Choose a running application in the list.

   This takes you to the overview of the application. Charts allow you to get a quick overview of the following metrics:
   - The number of HTTP requests processed by the Java application per hour over the last 24 hours
   - The maximum CPU consumption of the Java application per hour over the last 24 hours
5. To view the metrics for the processes, choose *Monitoring* > *Processes* in the navigation area.

   Alternatively, choose the link in the *Processes* tile of status summary.
6. Select a process to view the current state of its metrics.

   Details about two groups of metrics are shown – those registered by the platform (default) like CPU usage or Average Response Time and the custom ones registered by the user (user-defined). Furthermore, the listed custom metrics are on subaccount and application levels.
7. To view the history of monitoring metrics, depending on whether you want to view them on an application or process level, proceed as follows:
   - Application level - open the application whose history of metrics you want to see and choose *Monitoring* > *Application Monitoring* in the navigation area. All application processes, including those that are currently stopped, are visualized on the same charts so you can compare them. Furthermore, you can also see the history of the custom checks created on subaccount level.
   - Process level - on the *Application Monitoring* page, use the filter to choose a process. You can choose the filter value *All Processes* to display the metrics history of the whole application.

When you open the checks history, you can view graphic representations of the different checks, and zoom in when you click and drag horizontally or vertically to get further details. If you zoom in a graphic horizontally, all other graphics also zoom in to the same level of detail. Press `Shift` and drag to pan a graphic. Zoom out to the initial size with a double-click.

You can select different time intervals for viewing the checks. Depending on the selected interval, data is aggregated as follows:

- Last 12 or 24 hours - data is collected every minute.
- Last 7 days - data is aggregated from the average values for each 5 minutes.
- Last 30 days - data is aggregated from the average values for each hour.

You can also select a custom time interval when you are viewing history of checks. If you select an interval during which the application isn't running, the graphics won't contain any data.

8. To view the JMX checks created by the user on application level, choose the *Custom JMX Checks* tab.

   The subaccount level custom checks are not listed on this view.

Related Information

SAP BTP Cockpit
Alerting [page 779]
Availability Checks [page 732]
JMX Checks [page 738]

1.12.2 Availability Checks

Create an availability check for a Java or an SAP HANA XS application to track if the application is available and to receive alerts for it.

The availability check is one per Java or SAP HANA XS application and is executed every minute. You can configure an availability check for an application from the cockpit, from the console client, or with the Checks API.

According to the specified thresholds, you can view the following health states of the checks:

State	Description
CRITICAL	Your application isn't available or the response time is above the CRITICAL threshold.
WARNING	The response time of your application is above the WARNING threshold but not above the CRITICAL threshold.
OK	Your application has recovered from CRITICAL/WARNING state or the response time metric isn't above the WARNING threshold.

If your application isn't available or its response time is too high, you'll receive an alert notification, provided your application has started and you have configured alerting. The alert is triggered if the application isn't in

state OK for two consecutive checks. Furthermore, an alert is also triggered when the state of a check changes. For example, the state changes from CRITICAL/WARNING to OK. If you stop the application by yourself, you won't receive a notification as in this case alerting is suppressed and enabled once again when you start the application. However, this isn't valid for productive SAP HANA databases as you can't stop them. In this case, the availability check starts running at the moment you create it and won't stop until you delete it.

You may also set your availability check for Java applications on subaccount level using a relative URL. This means that each application started in your subaccount will immediately receive an availability check requesting `application_url/configured_relative_url`. This option is useful in case you start multiple instances of the same application (applications with the same relative health check URL) in your subaccount and allows you to configure this check only once for all of them. You can configure availability checks on subaccount level from the SAP BTP cockpit, the Neo console client, or with the Checks API. If there is a check configured on subaccount level and a check configured on application level, the one on the application level has higher priority. For example, if you have in your subaccount 10 applications with the `/health_check` relative URL and one multitenant application with the `/myapp/health_check` relative URL, you can configure an availability check on subaccount level for all applications and one availability check for the multitenant application to override the one on subaccount level.

Limitations

Availability monitoring in SAP BTP is done by running HTTP GET requests against URL provided by the application operator. The `http/https` ping is not parsing the response body, but it is relying only on the HTTP response code.

Currently, there are two limitations that need to be considered when designing your availability URL:

- The monitoring infrastructure does not support authorization for the checks. This means that you cannot pass user and password or client certificate when configuring the availability check. Therefore, you must design the availability URL without authentication or authorization. This will make sure that your application can be accessed in any case, the correct response code is returned (for example 200, 404, 500 and so on) and the response time is only from your application. If your application responds with 302, the ping will follow the redirect.

  > ⚠ Caution
  >
  > If you design the availability URL as a protected resource, the check will consider 401 and 403 response codes as 200 OK. Note that these response codes may come from Identity Authentication and not from your application, in case of an authenticated application.

  Currently, the response codes accepted by the `http/https` ping are **200**, **302**, **401** and **403**. This is done to cover all the different types of URLs that can be monitored. You need to make sure that if something does not work as expected, your application is not returning some of the above 4 codes as you will not get an alert.
- The monitoring infrastructure supports only one availability check per Java or SAP HANA XS application. This means that if you have multiple web applications deployed together as one application in your subaccount or application with multiple end points you want to check, you need to design one common availability URL to be able to monitor them all together. If one of the applications fails, you will get an alert and then you will have to check which one exactly is failing by opening the availability URL.

> ### → Recommendation
>
> We recommend that the response is a simple, plain HTML, just stating which web application is OK and which is not. It depends on the implementation of the availability URL whether it will just inform that a web application is available or it will also check whether it is working as expected. If you plan to develop and operate multiple applications in your subaccount, it is a good idea to have identical availability URLs for the different applications (for example, `/availability`). This will allow you to configure the availability check only once on subaccount level.

Sample output of an application which is OK:

```
HTTP RETURN CODE 200 OK
Purchasing - OK
Sales - OK
Registration - OK
IDP - OK
```

Sample output of an application that has problems:

```
HTTP RETURN CODE 500 INTERNAL SERVER ERROR
Purchasing - OK
Sales - no connectivity to backend
Registration - OK
IDP - OK
```

> ### ⚠ Caution
>
> Note that the availability URL designed according to the above recommendations is unprotected and can be accessed by everyone. We recommend not putting sensitive information about your application there (for example error stack traces).

Related Information

Alerting [page 779]
Configure Availability Checks for Java Applications from the Cockpit [page 735]
Configure Availability Checks for Java Applications from the Console Client [page 737]
Checks REST API for Java Applications [page 751]
Configure Availability Checks for SAP HANA XS Applications from the Cockpit [page 1006]
Configure Availability Checks for SAP HANA XS Applications from the Console Client [page 1007]
Checks REST API for SAP HANA XS Applications [page 777]
JMX Checks [page 738]

Availability Checks Commands

list-availability-check [page 1477]
create-availability-check [page 1383]
delete-availability-check [page 1401]

1.12.2.1 Configure Availability Checks for Java Applications from the Cockpit

In the cockpit, you can configure availability checks for the applications deployed in your subaccount. If you've configured an availability check on subaccount level, you can override it by creating one on application level.

Prerequisites

- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].
- You've deployed and started an application in your subaccount.

Context

In addition to the configuration of the availability check, you can configure alerting to receive alerts when your application is down. For this configuration, see the Related Information section.

Configure an Availability Check on Subaccount Level

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Choose *Monitoring* > *Java Custom Checks* in the navigation area for the subaccount.
3. Choose *Create Custom Check*.
4. Select the *Availability* type, specify the URI that is used for monitoring all the applications in the subaccount, and fill in values for warning and critical thresholds if you want them to be different from the default ones.

   > **i Note**
   >
   > The cockpit uses the following default thresholds:
   >
   > | Threshold | Default value |
   > |---|---|
   > | Warning | 50 seconds |
   > | Critical | 60 seconds |

5. Choose:

- *Create* when configuring a new check.
- *Update* when editing an existing check.

Configure an Availability Check on Application Level

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Choose *Applications* > *Java Applications* in the navigation area for the subaccount and then choose an application in the application list.
3. In the *Availability* panel, choose the following:
   - If an availability check isn't created on subaccount or application level, choose *Create Check*.
   - If an availability check is created on subaccount level but not on application level, choose *Override*.

     > **i Note**
     >
     > In such a case, a new availability check is created on application level, which has a priority over the one on subaccount level.

   - If an availability check already exists on application level, you can update it by choosing *Edit*.

     > **i Note**
     >
     > In such a case, the subaccout level availability check is ignored as the application one has a higher priority.

   You're allowed to create only one availability check per subaccount or application.
4. Select the URL you want to monitor from the dropdown list and fill in values for warning and critical thresholds if you want them to be different from the default ones.
5. Choose *Save*.

   Your availability check is configured. You can view your application's latest HTTP response code and response time as well as status icon showing whether your application is up or down.

Related Information

Alerting [page 779]
SAP BTP Cockpit
Availability Checks [page 732]

1.12.2.2 Configure Availability Checks for Java Applications from the Console Client

This topic shows how you can configure an availability check for your application. Furthermore, you can configure alerting to receive alerts when your application is down or responds slowly. For how to configure alerting, see the Related Information section.

Prerequisites

- You have deployed and started an application on the platform.
- You have set up the console client.
  For more information, see Set Up the Console Client [page 847].
- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Procedure

1. Open the command prompt and navigate to the folder containing `neo.bat and neo.sh (<SDK installation folder>/tools)`.
2. Create the availability check.

   Execute:

   ```
   neo create-availability-check -a mysubaccount -b myapp -u myuser -U /heartbeat -C 6 -W 4 --host hana.ondemand.com
   ```

   - Replace **"myapp"**, **"mysubaccount"** and **"myuser"** with the technical name of your subaccount, and the names of the application and user respectively.
   - The availability URL (**/heartbeat** in this case) is not provided by default by the platform. Replace it with a suitable URL that is already exposed by your application or create it for your application. Keep in mind the limitations for availability URLs, described in "Availability Checks" document (see Related Links below).
   - The check will trigger warnings **"-W 4"** if the response time is above **4** seconds and critical alerts **"-C 6"** if the response time is above 6 seconds or the application is not available.
   - Use the respective host for your subaccount type according to the region. For more information, see the *Related Information* section below.

   > **i Note**
   >
   > The availability check will be visible in the SAP BTP cockpit in around 2 minutes.

Related Information

Alerting [page 779]
Regions and Hosts Available for the Neo Environment [page 17]
Availability Checks [page 732]
JMX Checks [page 738]
Availability Checks Commands
list-availability-check [page 1477]
create-availability-check [page 1383]
delete-availability-check [page 1401]

1.12.3 JMX Checks

Registering JMX checks allows alerting on any metric that is based on JMX MBean attribute.

The checks support attributes that are `java.lang.String`, `java.lang.Number`, or `CompositeDataSupport`. When the attributes are `CompositeDataSupport`, the objects that are mapped to the keys again should be `java.lang.String` or `java.lang.Number`. Otherwise, an error will be thrown. For more information, see CompositeDataSupport .

The MBean can be registered either by the application runtime (for example, standard JVM MBeans like `java.lang:type=Memory`) or by the application itself (application-specific). The MBeans registered by the application runtime can be checked using the *jconsole* tool and connecting to the local server from the SDK.

You can set multiple JMX checks per application. They will be executed each minute.

According to the specified thresholds, you can view the following health states of the checks:

State	Description
CRITICAL	The JMX check fails due to an error in the MBean execution or the attribute value is above the defined CRITICAL threshold.
WARNING	The attribute value is above the defined WARNING threshold but not above the CRITICAL threshold.
OK	The attribute has recovered from CRITICAL/WARNING state or isn't above the WARNING threshold.

In case the JMX check fails due to an error in the MBean execution like, for example, wrong *ObjectName*, *Attribute*, *MBean* not registered, and so on, or due to exceeded threshold, you will receive an alert if you have configured alerting. The alert is triggered only after two consecutive failures of a JMX check. Furthermore, an alert is also triggered when the state of a check changes. For example, the state changes from CRITICAL/WARNING to OK.

You may also set JMX checks on subaccount level from the SAP BTP cockpit, the Neo console client, or with the Checks API. This means that each application started in your subaccount will immediately receive all the JMX checks configured on subaccount level in addition to the checks configured on the application level. If there is a check configured on subaccount level and a check configured on application level with one and the same name, the one on the application level has higher priority and only it will be assigned to the started application.

Related Information

Alerting [page 779]
Configure JMX Checks for Java Applications from the Cockpit [page 739]
Configure JMX Checks for Java Applications from the Console Client [page 741]
Checks REST API for Java Applications [page 751]
Availability Checks [page 732]
Add Custom Metrics to Java Apps by Using Third-Party Tools
JMX Checks Commands
list-jmx-checks [page 1492]
create-jmx-check [page 1390]
delete-jmx-check [page 1410]

1.12.3.1 Configure JMX Checks for Java Applications from the Cockpit

Configure a JMX check from the SAP BTP cockpit to monitor your Java application.

Prerequisites

- You've deployed and started a Java application on the platform.
- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Context

> **Note**
>
> The below procedures describes how you configure a custom JMX check on application level. If you want to configure one on subaccount level, go to *Monitoring* > *Java Custom Checks* in the SAP BTP cockpit and provide the object name, attribute, and composite key (when needed) as well as the additional properties for the MBean.

After you create the JMX check, you can monitor your Java application by the health state of the JMX metric produced from the created JMX check. To view the heath state of the JMX metric, use the *Processes* page in the cockpit.

You can also configure alerting to receive alert notifications for problematic JMX metrics. For more information, see the Related Information section. Note: The alert for a JMX check also includes the process ID.

Create a JMX Check on Application Level

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. In the cockpit, navigate to the started Java application you want to monitor.

   You can do this step by choosing the Java application under *Applications* > *Java Applications* or by navigating from the *Overview* page.
3. To create a JMX check, do one of the following:
   - Search for the MBean.
     Select *JMX Console* in the navigation area. For a selected process, search and select your MBean from the drop-down list. When you choose *Create Check* for the MBean on the *Attributes* tab, the MBean's object name, attribute, and composite key are populated automatically.
   - Directly provide the object name, attribute, and composite key (when needed) of the existing MBean. Select *Application Monitoring* or *Processes* in the navigation area and then choose *Create JMX Check*.

   In addition, you can specify the thresholds and the unit of measurement of the JMX metric to monitor your application with as well as what operation to be executed on your MBean for the check.
4. Choose *Create*.

   You can view the health state of your JMX metric, its current value, the thresholds, and the metric details on the *Processes* page.

Update a JMX Check on Application Level

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. In the cockpit, navigate to the started Java application you want to monitor.

   You can do this step by choosing the Java application under *Applications* > *Java Applications* or by navigating from the *Overview* page.
3. Select *Application Monitoring* in the navigation area and then *Custom JMX Checks* on the page.
4. In the custom JMX checks table, choose (Edit) for the respective check.
5. Update the object name, attribute, composite key, or the additional properties.

   > **Note**
   >
   > You can't update the name of a check.

   Choose *Update*.

Delete a JMX Check on Application Level

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. In the cockpit, navigate to the started Java application you want to monitor.

   You can do this step by choosing the Java application under *Applications* > *Java Applications* or by navigating from the *Overview* page.
3. Select *Application Monitoring* in the navigation area and then *Custom JMX Checks* on the page.
4. In the custom JMX checks table, choose (Delete) for the respective check.
5. Confirm the delete operation.

   The row for the check is removed from the table.

Related Information

Alerting [page 779]
JMX Checks [page 738]
Configure JMX Checks for Java Applications from the Console Client [page 741]
Checks REST API for Java Applications [page 751]
SAP BTP Cockpit

1.12.3.2 Configure JMX Checks for Java Applications from the Console Client

Configure a JMX check from the console client to monitor your Java application.

Prerequisites

- You have deployed and started an application on the platform.
- You have set up the console client.
  For more information, see Set Up the Console Client [page 847].
- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Context

After you create the JMX check, you can also configure alerting to receive alert notifications. For more information, see the Related Information section. Note: The alert for a JMX check also includes the process ID.

> **i** Note
>
> You can also update the JMX check from the console client by using the `create-jmx-check` command and the `--overwrite` parameter. To delete a JMX check, use the `delete-jmx-check` command. For the available JMX commands, see the Related Information section.

Furthermore, you can use the console client or the Checks API to configure JMX checks for your Java applications. See the Related Information section.

Procedure

1. Open the command prompt and navigate to the folder containing `neo.bat/sh (<SDK installation folder>/tools)`.
2. Create the JMX check.

   Execute:

   ```
   neo create-jmx-check -a mysubaccount -b myapp -u myuser -O myMBeanObjectName -A myMBeanAttributeName -n myCheckName -C myCriticalThreshold -W myWarningThreshold -U unit --host hana.ondemand.com
   ```

   - Replace **"myapp"**, **"mysubaccount"** and **"myuser"** with the technical name of your subaccount, and the names of the application and user respectively.
   - Replace **"myMBeanObjectName"** and **"myMBeanAttributeName"** with the attribute and object name of the MBean that you want to monitor. You can use existing standard MBean from the runtime (for example, standard JDK MBean like Catalina:type=ThreadPool,name=\"http-bio-8041\" and attribute like **currentThreadsBusy**) or your own MBean which should be part of your application and your application should register it in the MBean server. For more information about the attribute command, see "JMX Checks Commands" document in the *Related Information* section below.
   - Replace **"myCheckName"** with the name you want to see the check with in the cockpit.
   - Replace **"myWarningThreshold"** and **"myCriticalThreshold"** with a suitable threshold for the attribute you want to check. If the actual value is above the threshold, is out of the threshold range in case you use a range, or is a different string in case your metric has a string value, you'll receive a warning, respectively critical, notification. For more details how to set a threshold, see the *JMX Check Commands* document.
   - Replace **"unit"** with the unit you want to be displayed next to the value of your MBean attribute, for example MBs or ms.
   - Use the respective host for your subaccount type. For more information, see the *Related Information* section below.

Related Information

Alerting [page 779]
JMX Checks [page 738]
Configure JMX Checks for Java Applications from the Cockpit [page 739]
Checks REST API for Java Applications [page 751]
JMX Checks Commands
list-jmx-checks [page 1492]
create-jmx-check [page 1390]
delete-jmx-check [page 1410]

1.12.3.3 Use the JMX Console for JMX Operations

The JMX console available in the cockpit enables you to monitor and manage the performance of the JVM and your Java applications running on the platform.

Prerequisites

- You've started a Java application in your subaccount.
- The *readMonitoringData* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].
- To execute JMX operations, the *executeJMXOperation* scope has to be assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Context

The JMX console in the cockpit is based on the Java Management Extensions (JMX) specification. It exposes all the MBeans registered in the platform runtime and allows you to execute operations on them and view their attributes to monitor and manage the performance of the JVM and your applications. The MBeans visible in the JMX console are standard JVM MBeans, SAP-specific MBeans, and MBeans registered by your application runtime. The usage of some MBeans that can be dangerous in cloud environment is restricted.

In addition, you can create JMX checks from the JMX console to monitor your application and to send alert notifications.

> **i** Note
>
> This task is about how to run operations on MBeans. To learn how to create JMX checks from the JMX console, see Configure JMX Checks for Java Applications from the Cockpit [page 739].

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. In the cockpit, navigate to the started Java application you want to monitor.

   You can do this step by choosing the Java application under *Applications* > *Java Applications* or by navigating from the *Overview* page.
3. Open *Monitoring* > *JMX Console* in the navigation.
4. On the *JMX Console* page, select a process whose performance you want to check.

   This operation is possible only when multiple processes are running.
5. Browse for the MBean you want to monitor using the filter.
6. Expand the tree until you reach the MBean and choose it.

   The MBean attributes and operations are populated in the respective fields.
7. Depending on your needs, you can do the following:
   - Check the value of an attribute in the *Value* field.
   - Execute an MBean operation using (*Execute*) and check the results in the *Operation Results* section.

     > **i Note**
     >
     > A wait JMX operation times out after 60 seconds.

Related Information

SAP BTP Cockpit
JMX Checks [page 738]

1.12.4 Generate Heap and Thread Dumps

Use the JMX console to generate heap and thread dumps in order to analyze the performance of a Java process.

Prerequisites

- You've started a Java application in your subaccount.
- The *readMonitoringData* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].
- The *executeJMXOperation* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Context

You can check the state of the heap memory of a Java process by generating a heap dump. Furthermore, you can check the state of all the threads of a Java process by generating a thread dump. You perform these operations through the SAP BTP cockpit as you can download two files per a dump:

- The dump itself for your own analysis;
- A generated statistical analysis of the respective dump.

To generate a heap or thread dump, proceed as follows:

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. In the cockpit, navigate to the started Java application you want to monitor.

   You can do this step by choosing the Java application under *Applications* > *Java Applications* or by navigating from the *Overview* page.
3. Open *Monitoring* > *JMX Console* in the navigation.
4. On the *JMX Console* page, select a process whose performance you want to check.

   This operation is possible only when multiple processes are running.
5. Browse or search for the *com.sap.js* MBean.
6. Expand the tree and choose one of the following:
   - For a heap dump, choose the *HeapOperations* MBean.
   - For a thread dump, choose the *Threading* MBean.

   The MBean operations are populated in the respective fields.
7. Go to the *Operations* tab and execute the dump operation:
   - For a heap dump, execute the *dumpHeap* operation.
   - For a thread dump, execute the *dumpStacktrace* operation.

   The *result* row in the *Operation Results* section signifies that the execution of the dump has started.
8. Go to the *Logging* page to view and analyze the generated heap or thread dump.

   You can recognize the two most recent files for the generated dump by looking at the *Last Modified* date and time as well as the process ID. The *Type* column indicates if the downloadable file contains the dump itself or the analysis. To view the analysis in a browser, download the archive and choose `index.html`.

1.12.5 Metrics REST API for Java Applications

Use this REST API to get metrics for Java applications in the Neo environment.

Protection

The Metrics REST API is currently available with the following basic URI: `https://api.{host}/monitoring/v2`.

It's protected with OAuth 2.0 client credentials and is used for Java, HTML5, and SAP HANA XS applications as well as database systems in the Neo environment. Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149]. For more information about the format of the REST API, see Metrics API.

> **i Note**
>
> While you are creating the API client on the *Platform API* tab, select the *Monitoring Service* API with the *Read Monitoring Data* scope.

Overview

You can develop a custom application to request the states or the metric details for your Java applications and the applications' processes. Request the states or the metric details using the `GET` REST API calls.

> **Example**
>
> Use the following request to receive all the metrics for a Java application located in the Europe (Rot/Germany) region (with `hana.ondemand.com` host):
>
> ```
> https://api.hana.ondemand.com/monitoring/v2/accounts/<subaccount_technical_name>/apps/<application_name>/metrics
> ```

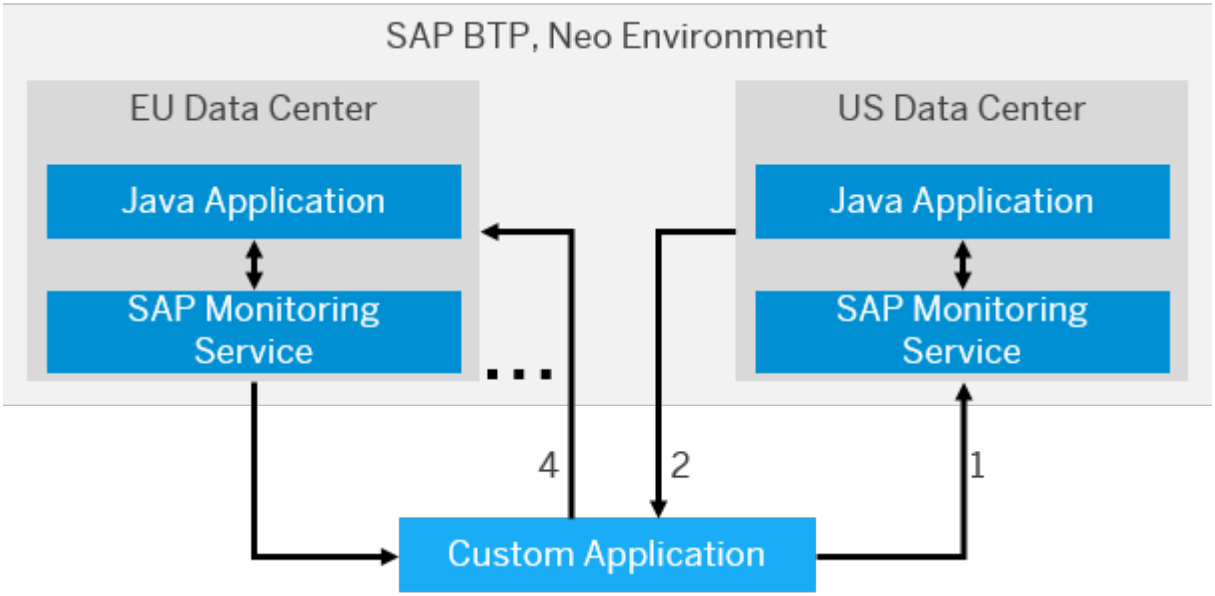
Benefits

You can use the REST API of SAP Monitoring service to perform the following actions:

- Implement elastic scaling for your Java applications.
  For more information, see Elastic Scaling on SAP BTP: How to Create a Simple Automatic Application Scaler in the Neo Environment.
- Retrieve and show the metrics for many Java applications located on different subaccounts and in different regions.
  For more information, see Use SAP Monitoring Service to Retrieve Metrics from Different Java Applications.

- Be notified of all critical metrics for many Java applications via e-mail, SMS, or another channel.
  For more information, see Use SAP Monitoring Service for Critical Notifications and Self-Healing of Java Applications.
- Take actions for application self-healing when critical metrics are received.
  You can set this up by using the lifecycle REST API. For more information, see Use SAP Monitoring Service for Critical Notifications and Self-Healing of Java Applications.

Process Flow



1. A custom application requests metrics for a Java application from SAP Monitoring service via a REST API call.
2. SAP Monitoring service sends back a JSON response with a status code 200 OK.
   The format of the REST API request specifies the metrics to be returned in the JSON response.
3. The custom application uses these metrics to perform operations.
4. The custom application requests metrics for other Java applications by repeating steps 1 to 3.

Related Information

Java ALM API documentation
SAP Monitoring Service Response for Java Applications [page 748]
Implement a Dashboard Application for Monitoring Java Apps
Implement a Notification Application for Monitoring Java Apps
Regions and Hosts Available for the Neo Environment [page 17]
Metrics REST API for HTML5 Applications [page 758]
Metrics REST API for SAP HANA XS [page 776]
Metrics REST API for Database Systems [page 775]

1.12.5.1 SAP Monitoring Service Response for Java Applications

Retrieve Java application metrics in a JSON format by performing a REST API request defined by the Metrics API.

JSON Response Parameters

Parameter	Value
`account`	The technical name of the subaccount as a string
`application`	The name of the application as a string
`state`	*OK*, *WARNING* (not used for a status metric), or *CRITICAL* This parameter specifies the state of the application, metric, or process.
`processes`	A list of processes. Each process contains the process ID, the state of the process, and the list of the metrics for that process.
`process`	The process ID as a string
`metrics`	A list of metrics. Each metric contains the following parameters: `name`, `state`, `value`, `unit`, `warningThreshold`, `errorThreshold`, `timestamp`, `output`, `metricType`, `min`, `max`.
`name`	The name of the metric as a string
`value`	The metric value as an integer
`unit`	The unit for the value as a string
`warningThreshold`	The warning threshold or the lower threshold as an integer
`errorThreshold`	The critical threshold or the upper threshold as an integer
`timestamp`	The time, in milliseconds, from January 1, 1970 to this call as a positive integer
`output`	A summary of the metric as a string
`min`	The minimum value as an integer.
`max`	The maximum value as an integer.

Parameter	Value
`metricType`	*status*, *performance*, or *rate* The state of each metric type based on the following metrics: • Status metric - *OK* or *CRITCAL* state of the metric is returned directly. • Performance metric - the returned value is compared with the predefined warning and critical thresholds. Values less than the warning threshold result in an *OK* state, values between the warning and the critical thresholds result in a *WARNING* state, and values greater than the critical threshold are considered as *CRITICAL*. • Rate metric - similar to the performance metric, but the returned value is calculated as a percent using the minimum and maximum values and compared with the warning and critical thresholds.

Example

The JSON response for Java application metrics may look like the following example:

```
[
   {
      "account": "mySubaccount",
      "application": "hello",
      "state": "Ok",
      "processes": [
         {
            "process": "bf061f611cc520f39839f2fa9e44813b2a20cdb7",
            "state": "Ok",
            "metrics": [
               {
                  "name": "Used Disc Space",
                  "state": "Ok",
                  "value": 43,
                  "unit": "%",
                  "warningThreshold": 90,
                  "errorThreshold": 95,
                  "timestamp": 1456408611000,
                  "output": "DISK OK - free space: / 4177 MB (54% inode=84%); /
var 1417 MB (74% inode=98%); /tmp 1845 MB (96% inode=99%);",
                  "metricType": "rate",
                  "min": 0,
                  "max": 8063
               },
               {
                  "name": "Requests per Minute",
                  "state": "Ok",
                  "value": 0,
                  "unit": "requests",
                  "warningThreshold": 0,
                  "errorThreshold": 0,
```

```
                  "timestamp": 1456408611000,
                  "output": "JMX OK - RequestsCountMin = 0 ",
                  "metricType": "performance",
                  "min": 0,
                  "max": 0
               },
               {
                  "name": "CPU Load",
                  "state": "Ok",
                  "value": 2,
                  "unit": "%",
                  "warningThreshold": 80,
                  "errorThreshold": 90,
                  "timestamp": 1456408611000,
                  "output": "OK CPUValue: 2 (W> 80, C> 90) ",
                  "metricType": "performance",
                  "min": 0,
                  "max": 0
               },
               {
                  "name": "Disk I/O",
                  "state": "Ok",
                  "value": 36386,
                  "unit": "B/s",
                  "warningThreshold": 1.0E7,
                  "errorThreshold": 1.5E7,
                  "timestamp": 1456408611000,
                  "output": "OK: DiskRead: 0 B/s (W> 10000000, C> 15000000)
DiskWrite: 36386 B/s (W> 10000000, C> 15000000)",
                  "metricType": "performance",
                  "min": 0,
                  "max": 0
               },
               {
                  "name": "OS Memory Usage",
                  "state": "Ok",
                  "value": 41,
                  "unit": "%",
                  "warningThreshold": 98,
                  "errorThreshold": 98,
                  "timestamp": 1456408611000,
                  "output": "OK: MemoryValue: 41 (W> 98, C> 98) ",
                  "metricType": "performance",
                  "min": 0,
                  "max": 0
               },
               {
                  "name": "Heap Memory Usage",
                  "state": "Ok",
                  "value": 8,
                  "unit": "%",
                  "warningThreshold": 0,
                  "errorThreshold": 0,
                  "timestamp": 1456408611000,
                  "output": "HeapMemoryUsage.used = 101 of 1224m MB ",
                  "metricType": "rate",
                  "min": 0,
                  "max": 1224
               },
               {
                  "name": "Average Response Time",
                  "state": "Ok",
                  "value": 0,
                  "unit": "ms",
                  "warningThreshold": 0,
                  "errorThreshold": 0,
                  "timestamp": 1456408611000,
                  "output": "JMX OK - AverageResponseTimeMin = 0ms ",
```

```
                    "metricType": "performance",
                    "min": 0,
                    "max": 0
                },
                {
                    "name": "Busy Threads",
                    "state": "Ok",
                    "value": 0,
                    "warningThreshold": 150,
                    "errorThreshold": 180,
                    "timestamp": 1456408611000,
                    "output": "JMX OK - currentThreadsBusy = 0 ",
                    "metricType": "performance",
                    "min": 0,
                    "max": 0
                }
            ]
        }
    ]
}
]
```


Related Information

Metrics REST API for Java Applications [page 746]
Monitoring Java Applications [page 729]

1.12.6 Checks REST API for Java Applications

Use the Checks API to retrieve, set, update, or delete custom checks for Java applications in the Neo environment.

Protection

This REST API is protected with OAuth 2.0 client credentials and is used for Java, SAP HANA XS, and HTML5 applications in the Neo environment. Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149]. For more information about the format of the REST APIs, see Checks API.

> **i Note**
>
> While you're creating the API client on the *Platform API* tab, select the *Monitoring Service* API with the *Manage Monitoring Configuration* scope.

Overview

You can use this API to configure availability and custom JMX checks for your Java applications:

- On subaccount level, you can manage the availability check or the custom JMX checks configured for all Java applications in the subaccount.
- On application level, you can manage the availability check or the custom JMX checks for your Java application.

Example

Setting a Custom JMX Check on Subaccount Level

Use the following POST request to set the custom JMX check `Compilation Time` for all Java applications in the subaccount in the Europe (Rot) region:

URI: `https://api.hana.ondemand.com/monitoring/checks/v1/accounts/<subaccount_technical_name>/types/jmx-check`

Body:

```
[
  {
    "checkName": "Compilation Time",
    "typeName": "jmx-check",
    "object-name": "java.lang:type=Compilation",
    "attribute": "TotalCompilationTime",
    "warning": "120000",
    "critical": "130000"
  }
]
```

Example

Updating an Availability Check on Application Level

Use the following PUT request to update an availability check with endpoint `/healthcheck` for your Java application in the US East (Ashburn) region:

URI: `https://api.us1.hana.ondemand.com/monitoring/checks/v1/accounts/<subaccount_technical_name>/apps/<application_name>/types/availability-check`

Body:

```
[
  {
    "uri": "/healthcheck",
    "warning": "30",
    "critical": "40"
  }
]
```

Related Information

Availability Checks [page 732]

JMX Checks [page 738]
Regions and Hosts Available for the Neo Environment [page 17]
Checks REST API for HTML5 Applications [page 759]
Checks REST API for SAP HANA XS Applications [page 777]

1.13 Monitoring HTML5 Applications

To monitor your HTML5 application, create custom checks for it by specifying your own metrics. Furthermore, configure alert notifications for any changes to the states of the configured checks.

Overview

For the configuration of the custom checks, specify an application URL endpoint to monitor. Register these checks for HTML5 applications from the SAP BTP cockpit or with the Checks API as the checks are executed every minute. You also process the checks by viewing their metrics in the SAP BTP cockpit or by using the Metrics REST API. With the Metrics API, you can receive the current state of the configured checks as well as their metrics.

Furthermore, you can configure alerting when the health state of the custom check has changed. For example, if the state has changed from OK to WARNING or vice versa.

> **!** Restriction
>
> Metrics are also retrieved for a stopped application, and you see a metrics history even for the period when the monitored HTML5 application was stopped.

Authentication

Only basic authentication is supported for the URL. You can configure basic authentication by passing a username and password if your URL is protected. However, if you decide to use no authentication, you must design the custom check URL with no authentication or authorization.

View Metrics of Custom Checks for an HTML5 Application [page 754]
: In the cockpit, you can view the current state and metrics of a selected application. You can also view the metrics history for an application to examine performance trends over a different period of time, or to investigate any problems with it that may arise.

Configure Custom Checks for HTML5 Applications [page 755]
: Configure custom checks to monitor your application. As a result, in the SAP BTP cockpit you can view the current state and metrics of the configured checks or their metrics history.

Metrics REST API for HTML5 Applications [page 758]
: Use the REST API to get metrics for your HTML5 applications that are running on SAP BTP in the Neo environment.

Checks REST API for HTML5 Applications [page 759]
: Use the Checks API to retrieve, set, update, or delete custom checks for HTML5 applications in the Neo environment.

Related Information

Alerting [page 779]

1.13.1 View Metrics of Custom Checks for an HTML5 Application

In the cockpit, you can view the current state and metrics of a selected application. You can also view the metrics history for an application to examine performance trends over a different period of time, or to investigate any problems with it that may arise.

Prerequisites

- An HTML5 application is deployed and started in your subaccount.
- The *readMonitoringData* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].
- Custom checks are created to monitor the application.

Context

According to the specified thresholds, you can view the following health states of the checks:

State	Description
CRITICAL	The application is unavailable, the response code is greater than or equal to 300, or the response time is above the CRITICAL threshold.
WARNING	The the response time of your application is above the WARNING threshold but not above the CRITICAL threshold.
OK	The application has recovered from CRITICAL/WARNING states or the response time metric isn't above the WARNING threshold.

When the state of a check changes, you receive an alert notification, provided your application has started and you have configured alerting.

> **Note**
>
> You can also retrieve all the checks for an HTML5 application by using the Checks API.

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Choose *Applications* > *HTML5 Applications* in the navigation area for the subaccount.
3. Choose a started application from the list.

   You navigate to the overview page of the application.
4. Select *Monitoring* > *Application Monitoring*.

   You see details about the current custom metrics in a table.
5. Scroll down to the *Timeline* section to see the history of metrics.

   When you open the checks history, you can view graphic representations for each of the different checks, and zoom in to see additional details. If you zoom in a graphic horizontally, all other graphics also zoom in to the same level of detail. Press `Shift` and drag to pan a graphic. Zoom out to the initial size by double-clicking.

   You can select different periods for each check. Depending on the interval you select, data is aggregated as follows:
   - Last 12 or 24 hours - data is collected every minute.
   - Last 7 days - data is aggregated from the average values for each 10 minutes.
   - Last 30 days - data is aggregated from the average values for each hour.

   You can also select a custom time interval for viewing check history.

Related Information

Alerting [page 779]
SAP BTP Cockpit
Checks REST API for HTML5 Applications [page 759]

1.13.2 Configure Custom Checks for HTML5 Applications

Configure custom checks to monitor your application. As a result, in the SAP BTP cockpit you can view the current state and metrics of the configured checks or their metrics history.

Prerequisites

- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].
- You know the warning and critical thresholds for your custom check.
  You're required to provide these thresholds for the application response time in seconds when you create a custom check.

Context

You can configure custom checks either from the SAP BTP cockpit or with the Checks API.

Furthermore, receive alerts for any changes to the states of these metrics. For more information, see the Related Information section.

> **i Note**
>
> Any change related to the creation or update of a custom check is reflected after 5 minutes.

Create a Custom Check

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. From the navigation area, choose *Applications* > *HTML5 Applications*.
3. Choose a running application from the list.
4. Select *Application Monitoring* and choose *Create Custom Check*.
   a. Specify a name for the check.

      You can't change the name once you've created a check.
   b. Provide the URL and the threshold values.

      The URL is the endpoint of the application you would like to monitor, and the thresholds are for the application response time in seconds. By default, the SAP BTP cockpit provides the endpoint URL of the HTML5 application.
   c. If your URL is protected with basic authentication, provide a username and password.
   d. Choose *Create*.

      You can view the state of your custom check, its current value, the thresholds, and the returned status code.

Update a Custom Check

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Choose *Applications* > *HTML5 Applications* in the navigation area of the subaccount.
3. Choose a running application from the list.
4. Select *Application Monitoring* in the navigation area, and in the custom checks table, choose (Edit) for the respective custom check.
5. Update the endpoint URL, the thresholds, and the authentication type.

   > **i Note**
   >
   > You can't update the name of a check.

   Choose *Update*.

Delete a Custom Check

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Choose *Applications* > *HTML5 Applications* in the navigation area of the subaccount.
3. Choose a running application from the list.
4. Select *Application Monitoring* in the navigation area, and in the custom checks table, choose (Delete) for the respective custom check.
5. Confirm the delete operation.

   The row for the check is removed from the table.

Related Information

Alerting [page 779]
SAP BTP Cockpit
Checks REST API for HTML5 Applications [page 759]

1.13.3 Metrics REST API for HTML5 Applications

Use the REST API to get metrics for your HTML5 applications that are running on SAP BTP in the Neo environment.

Protection

The Metrics REST API is protected with OAuth 2.0 client credentials. Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149]. For more information about the format of the REST API, see Metrics API.

> **i Note**
>
> When you create the API client on the *Platform API* tab, select the *Monitoring Service* API with the *Read Monitoring Data* scope.

Overview

Request the states or the metric details for your HTML5 applications by using `GET` REST API calls.

> **Example**
>
> Use the following request to receive the state for an HTML5 application located in the Europe (Rot/Germany) region (with `hana.ondemand.com` host):
>
> ```
> https://api.hana.ondemand.com/monitoring/v2/accounts/<subaccount_technical_name>/
> html5/apps/<html5_app_name>/state
> ```

> **Example**
>
> Use the following request to receive all the metrics for an HTML5 application located in the US West (Chandler) region (with `us2.hana.ondemand.com` host):
>
> ```
> https://api.us2.hana.ondemand.com/monitoring/v2/accounts/
> <subaccount_technical_name>/html5/apps/<html5_app_name>/metrics
> ```

1.13.4 Checks REST API for HTML5 Applications

Use the Checks API to retrieve, set, update, or delete custom checks for HTML5 applications in the Neo environment.

Protection

This REST API is protected with OAuth 2.0 client credentials and is used for Java, SAP HANA XS, and HTML5 applications in the Neo environment. Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149]. For more information about the format of the REST APIs, see Checks API.

> **i Note**
>
> While you're creating the API client on the *Platform API* tab, select the *Monitoring Service* API with the *Manage Monitoring Configuration* scope.

Overview

You can use this API to configure custom checks for your HTML5 applications. Furthermore, you manage a particular check by specifying the check's name as a parameter or in the request's body.

> **Example**
>
> Setting a Custom Check
>
> Use the following POST request to set the custom check `Backend available` for an HTML5 application in the Europe (Rot) region:
>
> URI: `https://api.hana.ondemand.com/monitoring/checks/v1/accounts/<subaccount_technical_name>/html5/apps/<html5_app_name>`
>
> Body:

```
[
  {
    "checkName": "Backend available",
    "url": "https://somenewappurl.com/somedestination",
    "warning": "30",
    "critical": "40"
  }
]
```

> **Example**
>
> Updating a Custom Check

Use the following PUT request to update the thresholds for the custom check `Backend available`:

URI: `https://api.hana.ondemand.com/monitoring/checks/v1/accounts/<subaccount_technical_name>/html5/apps/<html5_app_name>`

Body:

```
[
  {
    "checkName": "Backend available",
    "url": "https://somenewappurl.com/somedestination",
    "warning": "40",
    "critical": "50"
  }
]
```

Related Information

Monitoring HTML5 Applications [page 753]
Regions and Hosts Available for the Neo Environment [page 17]
Checks REST API for Java Applications [page 751]
Checks REST API for SAP HANA XS Applications [page 777]

1.14 Monitoring Database Systems

To monitor whether your deployed SAP HANA XS application is up and running, you can register an availability check for it from the SAP BTP cockpit, from the console client, or with the Checks API.

In addition, you can configure alert notification if the application goes down. For the alerting configuration, you use the console client or the Alerting Channels REST API. Furthermore, you can configure alert notifications for the database systems in a specific subaccount. In such a case, you configure the alerting on subaccount level. As a result, SAP Monitoring service sends an alert notification when any type of a database system in the subaccount goes down.

Moreover, you can create custom checks to monitor the status or performance of the XS application by using the SAP BTP cockpit or the Checks API. You can view the metrics of an XS application or a database system of any type by using the SAP BTP cockpit or you can retrieve these metrics with the Metrics REST API.

View Monitoring Metrics of a Database System [page 761]
In the cockpit, you can view the current metrics of a selected database system to get information about its health state. You can also view the metrics history of a productive database to examine the performance trends of your database over different intervals of time or investigate the reasons that have led to problems with it. You can view the metrics for all types of databases.

View Monitoring Metrics of an SAP HANA XS Application [page 766]
In the SAP BTP cockpit, you can view the history of custom checks to help you monitor your SAP HANA XS application.

Availability Checks [page 767]

Create an availability check for a Java or an SAP HANA XS application to track if the application is available and to receive alerts for it.

Custom HTTP Checks [page 772]
: You can create custom checks for an SAP HANA XS application, and you can configure alert recipients for these checks. Alerts are sent to the configured recipients when the state of a check changes.

Metrics REST API for Database Systems [page 775]
: Use the REST API to get metrics for your database systems that are running in the Neo environment.

Metrics REST API for SAP HANA XS [page 776]
: Use the REST API to get metrics for your SAP HANA XS applications and instances that are running in the Neo environment.

Checks REST API for SAP HANA XS Applications [page 777]
: Use the Checks API to retrieve, set, update, or delete custom checks for HTML5 applications in the Neo environment.

Related Information

Alerting [page 779]
Monitoring Java Applications [page 729]

1.14.1 View Monitoring Metrics of a Database System

In the cockpit, you can view the current metrics of a selected database system to get information about its health state. You can also view the metrics history of a productive database to examine the performance trends of your database over different intervals of time or investigate the reasons that have led to problems with it. You can view the metrics for all types of databases.

Prerequisites

The *readMonitoringData* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Context

> **i Note**
>
> You can also retrieve the current metrics of a database system with the Metrics API.

Default Metrics of a Database System
All database systems include these default metrics. Other database-specific metrics may also be available.

Metric	Value	Execution Frequency
CPU Load	The percentage of the CPU that is used on average over the last minute.	This metric is updated every minute. An alert is triggered when 2 consecutive checks with an interval of 1 minute aren't in an OK state.
Disk I/O	The number of bytes per second that are currently being read or written to the disc.	This metric is updated every minute. An alert is triggered when 5 consecutive checks with an interval of 1 minute aren't in an OK state.
Network Ping	The percentage of packets that are lost to the database host.	This metric is updated every minute. An alert is triggered when 2 consecutive checks with an interval of 1 minute aren't in an OK state.
OS Memory Usage	The percentage of the operating system memory that is currently being used.	This metric is updated every minute. An alert is triggered when 2 consecutive checks with an interval of 1 minute aren't in an OK state.
Used Disc Space	The percentage of the local discs of the operating system that is currently being used. **i Note** If this metric is in a critical state, try restarting the database system. If the restart doesn't work, check the troubleshooting documentation. See the *Related Information* section.	This metric is updated every minute. An alert is triggered when 5 consecutive checks with an interval of 1 minute aren't in an OK state.

Default Metrics of an SAP HANA System

Metric	Value	Execution Frequency
HANA DB Availability	• *OK* - the database is reachable from our central admin component via JDBC. • *Critical* - either the database is down or overloaded, or there's a network issue.	This metric is updated every minute. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.

Metric	Value	Execution Frequency
HANA DB Alerting Availability	• *OK* - alerts can be retrieved from the SAP HANA system. • *Critical* - alerts can't be retrieved as there's no connection to the database. This also implies that any other visible metric may be outdated. This metric depends on the HANA DB Availability metric.	This metric is updated every minute. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Compile Server	• *OK* - the compiler server is running on the SAP HANA system. • *Critical* - the compile server crashed or was otherwise stopped. The service should recover automatically. If this doesn't work, a restart of the system might be necessary.	This metric is updated every 10 minutes. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Backup Volumes Availability	• *OK* - the backup volumes are available. • *Critical* - the backup volumes aren't available.	This metric is updated every 15 minutes. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Data Backup Age	• *OK* - the age of the last data backup is below the critical threshold. • *Critical* - the age of the last data backup is above the critical threshold.	This metric is updated every 24 hours. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Data Backup Exists	• *OK* - the data backup exists. • *Critical* - no data backup exists.	This metric is updated every 24 hours. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Data Backup Successful	• *OK* - the last data backup was successful. • *Critical* - the last data backup wasn't successful.	This metric is updated every 24 hours. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Log Backup Successful	• *OK* - the last log backup was successful. • *Critical* - the last log backup failed.	This metric is updated every 10 minutes. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.

Metric	Value	Execution Frequency
HANA DB Service Memory Usage	• *OK* - no server is running out of memory. • *Critical* - a service is causing an out of memory error. See SAP Note 1900257.	This metric is updated every 5 minutes. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA XS Availability	• *OK* - XSEngine accepts HTTPS connections. • *Critical* - XSEngine doesn't accept HTTPS connections.	This metric is updated every minute. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA Dump Files Count	• *OK* - No dump files exist. • *Warning* - Up to 20 dump files exist. • *Critical* - More than 20 dump files exist. Try to analyze the dump files. **Note** If you're still having issues, check the troubleshooting documentation. See the *Related Information* section.	The metric is updated every hour. An alert is triggered when a check isn't in an OK state.

Default Metrics of an SAP ASE System

Metric	Value	Execution Frequency
Sybase ASE Availability	• *OK* - the database is reachable from our central admin component via JDBC. • *Critical* - either the database is down or overloaded, or there's a network issue.	This metric is updated every minute. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
Sybase ASE Long Running Trans	• *OK* - a transaction is running for up to an hour. • *Warning* - a transaction is running for more than an hour. • *Critical* - a transaction is running for more than 13 hours.	This metric is updated every 2 minutes. An alert is triggered when a consecutive check with an interval of 1 minute isn't in an OK state.
Sybase ASE HADR Fm State	FaultManager is a component for highly available (HA) SAP ASE systems that triggers a failover in case the primary node isn't working. • *OK* - FaultManager for a system that is set up as an HA system is running properly. • *Critical* - FaultManager isn't working properly and the failover doesn't work.	This metric is updated every 2 minutes. An alert is triggered when a consecutive check with an interval of 1 minute isn't in an OK state.

Metric	Value	Execution Frequency
Sybase ASE HADR Latency	• *OK* - the latency for the HA replication path is less than or equal to 10 minutes. • *Warning* - the latency is greater than 10 minutes. • *Critical* - the latency is greater than 20 minutes. A high latency might lead to data loss if there's a failover.	This metric is updated every 2 minutes. An alert is triggered when a consecutive check with an interval of 1 minute isn't in an OK state.
Sybase ASE HADR Primary State	• *OK* - the primary host of a system that is set up as HA system is running fine. • *Critical* - the primary host isn't running properly.	This metric is updated every 2 minutes. An alert is triggered when a consecutive check with an interval of 1 minute isn't in an OK state.
Sybase ASE HADR Standby State	• *OK* - the secondary or standby host of a system that is set up as HA system is running properly. • *Critical* - the secondary or standby host isn't running properly.	This metric is updated every 2 minutes. An alert is triggered when a consecutive check with an interval of 1 minute isn't in an OK state.

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Navigate to the *Database Systems* page either by choosing *SAP HANA / SAP ASE* > *Database Systems* from the navigation area or from the *Overview* page.

   All database systems available in the selected subaccount are listed with their details, including the database version and state, and the number of associated databases.
3. Choose the entry for the relevant database system in the list.
4. Choose *Monitoring* from the navigation area to get detailed information about the current state and the history of metrics for the selected database system.

   When you open the checks history, you can view graphic representations for each of the different checks, and zoom in to see additional details. If you zoom in a graphic horizontally, all other graphics also zoom in to the same level of detail. Press `Shift` and drag to pan a graphic. Zoom out to the initial size by double-clicking.

   You can select different periods for each check. Depending on the interval you select, data is aggregated as follows:
   - Last 12 or 24 hours - data is collected every minute.
   - Last 7 days - data is aggregated from the average values for each 10 minutes.
   - Last 30 days - data is aggregated from the average values for each hour.

   You can also select a custom time interval for viewing check history.

Related Information

SAP BTP Cockpit
Metrics REST API for Database Systems [page 775]

1.14.2 View Monitoring Metrics of an SAP HANA XS Application

In the SAP BTP cockpit, you can view the history of custom checks to help you monitor your SAP HANA XS application.

Prerequisites

- The *readMonitoringData* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].
- An SAP HANA XS application is deployed and started in your subaccount.

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Choose *Applications* > *HANA XS Applications* in the navigation area of the subaccount.
3. Select an application from the list.
4. In the *Application Details* panel, you can view the history of the custom checks.

   When you open the checks history, you can view graphic representations for each of the different checks, and zoom in to see additional details. If you zoom in a graphic horizontally, all other graphics also zoom in to the same level of detail. Press `Shift` and drag to pan a graphic. Zoom out to the initial size by double-clicking.

   You can select different periods for each check. Depending on the interval you select, data is aggregated as follows:
   - Last 12 or 24 hours - data is collected every minute.
   - Last 7 days - data is aggregated from the average values for each 10 minutes.
   - Last 30 days - data is aggregated from the average values for each hour.

   You can also select a custom time interval for viewing check history.

1.14.3 Availability Checks

Create an availability check for a Java or an SAP HANA XS application to track if the application is available and to receive alerts for it.

The availability check is one per Java or SAP HANA XS application and is executed every minute. You can configure an availability check for an application from the cockpit, from the console client, or with the Checks API.

According to the specified thresholds, you can view the following health states of the checks:

State	Description
CRITICAL	Your application isn't available or the response time is above the CRITICAL threshold.
WARNING	The response time of your application is above the WARNING threshold but not above the CRITICAL threshold.
OK	Your application has recovered from CRITICAL/WARNING state or the response time metric isn't above the WARNING threshold.

If your application isn't available or its response time is too high, you'll receive an alert notification, provided your application has started and you have configured alerting. The alert is triggered if the application isn't in state OK for two consecutive checks. Furthermore, an alert is also triggered when the state of a check changes. For example, the state changes from CRITICAL/WARNING to OK. If you stop the application by yourself, you won't receive a notification as in this case alerting is suppressed and enabled once again when you start the application. However, this isn't valid for productive SAP HANA databases as you can't stop them. In this case, the availability check starts running at the moment you create it and won't stop until you delete it.

You may also set your availability check for Java applications on subaccount level using a relative URL. This means that each application started in your subaccount will immediately receive an availability check requesting `application_url/configured_relative_url`. This option is useful in case you start multiple instances of the same application (applications with the same relative health check URL) in your subaccount and allows you to configure this check only once for all of them. You can configure availability checks on subaccount level from the SAP BTP cockpit, the Neo console client, or with the Checks API. If there is a check configured on subaccount level and a check configured on application level, the one on the application level has higher priority. For example, if you have in your subaccount 10 applications with the `/health_check` relative URL and one multitenant application with the `/myapp/health_check` relative URL, you can configure an availability check on subaccount level for all applications and one availability check for the multitenant application to override the one on subaccount level.

Limitations

Availability monitoring in SAP BTP is done by running HTTP GET requests against URL provided by the application operator. The `http/https` ping is not parsing the response body, but it is relying only on the HTTP response code.

Currently, there are two limitations that need to be considered when designing your availability URL:

- The monitoring infrastructure does not support authorization for the checks. This means that you cannot pass user and password or client certificate when configuring the availability check. Therefore, you must design the availability URL without authentication or authorization. This will make sure that your application can be accessed in any case, the correct response code is returned (for example 200, 404, 500 and so on) and the response time is only from your application. If your application responds with 302, the ping will follow the redirect.

  > ⚠ Caution
  >
  > If you design the availability URL as a protected resource, the check will consider 401 and 403 response codes as 200 OK. Note that these response codes may come from Identity Authentication and not from your application, in case of an authenticated application.

  Currently, the response codes accepted by the `http/https` ping are **200**, **302**, **401** and **403**. This is done to cover all the different types of URLs that can be monitored. You need to make sure that if something does not work as expected, your application is not returning some of the above 4 codes as you will not get an alert.
- The monitoring infrastructure supports only one availability check per Java or SAP HANA XS application. This means that if you have multiple web applications deployed together as one application in your subaccount or application with multiple end points you want to check, you need to design one common availability URL to be able to monitor them all together. If one of the applications fails, you will get an alert and then you will have to check which one exactly is failing by opening the availability URL.

  > → Recommendation
  >
  > We recommend that the response is a simple, plain HTML, just stating which web application is OK and which is not. It depends on the implementation of the availability URL whether it will just inform that a web application is available or it will also check whether it is working as expected. If you plan to develop and operate multiple applications in your subaccount, it is a good idea to have identical availability URLs for the different applications (for example, `/availability`). This will allow you to configure the availability check only once on subaccount level.

Sample output of an application which is OK:

```
HTTP RETURN CODE 200 OK
Purchasing - OK
Sales - OK
Registration - OK
IDP - OK
```

Sample output of an application that has problems:

```
HTTP RETURN CODE 500 INTERNAL SERVER ERROR
Purchasing - OK
Sales - no connectivity to backend
Registration - OK
IDP - OK
```

> ⚠ Caution
>
> Note that the availability URL designed according to the above recommendations is unprotected and can be accessed by everyone. We recommend not putting sensitive information about your application there (for example error stack traces).

Related Information

Alerting [page 779]
Configure Availability Checks for Java Applications from the Cockpit [page 735]
Configure Availability Checks for Java Applications from the Console Client [page 737]
Checks REST API for Java Applications [page 751]
Configure Availability Checks for SAP HANA XS Applications from the Cockpit [page 1006]
Configure Availability Checks for SAP HANA XS Applications from the Console Client [page 1007]
Checks REST API for SAP HANA XS Applications [page 777]
JMX Checks [page 738]
Availability Checks Commands
list-availability-check [page 1477]
create-availability-check [page 1383]
delete-availability-check [page 1401]

1.14.3.1 Configure Availability Checks for SAP HANA XS Applications from the Cockpit

In the SAP BTP cockpit, you can configure availability checks for the SAP HANA XS applications running on your productive SAP HANA database system.

Prerequisites

- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].
- You have deployed and started an SAP HANA XS application in your subaccount.

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Choose *Applications* > *HANA XS Applications* in the navigation area of the subaccount.
3. Select an application from the list, and in the *Application Details* panel, configure the check.
    - To create a new check, choose *Create Availability Check*.
    - To view a created availability check, choose *Details*.
    - To update a created availability check, choose *Details* > *Edit*.
    - To delete a created availability check, choose *Details* > *Delete*.

4. In the dialog that appears, select or enter the URL you want to monitor and fill in values for warning and critical thresholds if you want them to be different from the current ones. The default values are 50 seconds for a warning threshold and 60 seconds for a critical threshold.
5. Save your configuration.

   When your availability check is created, you can view your application's latest HTTP response code and response time as well as a status icon showing whether your application is up or down. If you want to receive alerts when your application is down, you have to configure alerting.

Related Information

Alerting [page 779]
SAP BTP Cockpit
Availability Checks [page 767]
Configure Availability Checks for SAP HANA XS Applications from the Console Client [page 1007]

1.14.3.2 Configure Availability Checks for SAP HANA XS Applications from the Console Client

In the console client, you can configure an availability check for your SAP HANA XS application and subscribe recipients to receive alert e-mail notifications when it is down or responds slowly. For how to configure alerting, see the Related Information section.

Prerequisites

- You have a productive SAP HANA database on the platform.
  For more information, see Using an SAP HANA XS Database System [page 999].
- You have set up the console client.
  For more information, see Set Up the Console Client [page 847].
- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Procedure

1. Open the command prompt and navigate to the folder containing `neo.bat/sh (<SDK installation folder>/tools)`.
2. Create the availability check.

Execute:

```
neo create-availability-check -a mysubaccount -b myhana:myhanaxsapp -u myuser
-U /heartbeat.xsjs -C 6 -W 4 --host hana.ondemand.com
```

- Replace **"mysubaccount"**, **"myhana:myhanaxsapp"** and **"myuser"** with the technical name of your subaccount, and the names of the productive SAP HANA database, application, and user respectively.
- The availability URL (**/heartbeat.xsjs** in this case) is not provided by default by the platform. Replace it with a suitable URL that is already exposed by your SAP HANA XS application or create it. Keep in mind the limitations for availability URLs. For more information, see Availability Checks [page 767].

  > **i Note**
  >
  > In case you want to create an availability check for a protected SAP HANA XS application, you need to create a subpackage, in which to create an `.xsaccess` file with the following content:
  >
  > ```
  > {
  >     "exposed": true,
  >     "authentication": null,
  >     "authorization": null
  > }
  > ```
  >
  > In the SAP HANA XS Administration Tool (https://<productive SAP HANA name><subaccount>.<host>/sap/hana/xs/admin/), find the subpackage and activate public access to it. After this, you will be able to monitor the resources on and under the subpackage level, that is, you will be able to create an availability check for your protected application. Note that in the availability URL, you have to provide the path to the resource in the subpackage that you will be monitoring, for example **/heartbeat.xsjs**.

- The check will trigger warnings **"-W 4"** if the response time is above **4** seconds and critical alerts **"-C 6"** if the response time is above 6 seconds or the application is not available.
- Use the respective host for your subaccount type.

Related Information

Alerting [page 779]
Configure Availability Checks for SAP HANA XS Applications from the Cockpit [page 1006]
Regions and Hosts Available for the Neo Environment [page 17]

Availability Checks Commands

list-availability-check [page 1477]
create-availability-check [page 1383]
delete-availability-check [page 1401]

1.14.4 Custom HTTP Checks

You can create custom checks for an SAP HANA XS application, and you can configure alert recipients for these checks. Alerts are sent to the configured recipients when the state of a check changes.

Overview

You can configure multiple custom HTTP checks for an SAP HANA XS application. For this configuration, you need to have the URL endpoint for the custom check that returns a JSON object with the value to be checked. You configure these checks for SAP HANA XS applications from the SAP BTP cockpit or through the Checks REST API, and the checks are executed every minute. Consequently, you process the checks in the SAP BTP cockpit or by using the Metrics REST API. With the REST API, you can receive the current state of the configured checks as well as the details. See Related Information.

The following types of checks are supported:

- *status*
  For a *status* check, SAP Monitoring service uses the provided endpoint and evaluates the returned HTTP status code. A return code of *200* indicates an OK state. A return code of anything other than 200 indicates a CRITICAL state. Moreover, the output is parsed from the returned JSON and shown even if the check is in a CRITICAL state. If parsing is unsuccessful, the entire response is shown.
- *performance*
  For a *performance* check, SAP Monitoring service compares the returned value with the predefined warning and critical thresholds. Values less than the warning threshold result in an OK state, values between the warning and the critical thresholds result in a WARNING state, and values greater than the critical threshold are considered as CRITICAL.

If a *status* check is in a CRITICAL state, or if the performance check is in a WARNING or CRITICAL state, you receive an alert notification, provided your application has started and you've configured alerting. There are the following types of notifications:

Notification	Description
CRITICAL	Your application is unavailable, or the *status* and *performance* checks are in a CRITICAL state.
WARNING	The performance check is in a WARNING state.
OK	Your application has recovered from a CRITICAL/WARNING state.

Limitation

The monitoring infrastructure doesn't support authorization for the checks. This means that you can't pass user and password information, or a client certificate when configuring the check. Therefore, you must design the custom check URL without authentication or authorization. This ensures that your application can be accessed in any case, the correct response code is returned (for example, 200, 404, 500, and so on), and the response is only from your application.

> ⚠ Caution
>
> If you design the custom check URL as a protected resource, the check may receive 401 and 403 response codes. These response codes may come from Identity Authentication and not from your application, in case of an authenticated application.

Related Information

Alerting [page 779]
Configure Custom Checks for SAP HANA XS Applications [page 773]
Metrics REST API for SAP HANA XS [page 776]
Checks REST API for SAP HANA XS Applications [page 777]

1.14.4.1 Configure Custom Checks for SAP HANA XS Applications

You configure status and performance checks to monitor the state of your application. Furthermore, you can configure alerting to receive alerts for any changes to the states of these checks. For how to configure alerting, see the Related Information section.

Prerequisites

- You have the URL endpoint for the application to monitor. The URL endpoint has to return a JSON object in the following format:

  > ‹≡› Sample Code
  >
  ```
  {
  "value": "<number value to be used by the check>"
  }
  ```

- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Create a Custom Check

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Choose *Applications* > *HANA XS Applications* in the navigation area of the subaccount.
3. Select an application from the list, and in the *Application Details* panel, choose *Create Custom Check*.
4. In the dialog that appears, select the type of the check as one of the following:
   - *status*
     For a status check, provide the name and the URL endpoint.
     By default, the URL field displays the base URL for the SAP HANA XS application. You're supposed to extend or update this URL to match with your application endpoint.
   - *performance*
     For a performance check, provide the warning and critical thresholds in addition to the name and the URL endpoint.
     You can optionally provide the unit of measurement for the metric, which is displayed in the custom checks table.

   Choose *Create*.

   Your custom check is created, and you can view the following data:

   - The state of your custom check
   - The retuned endpoint value
   - In a performance check, the thresholds.

Update a Custom Check

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Choose *Applications* > *HANA XS Applications* in the navigation area of the subaccount.
3. Select an application from the list, and in the custom checks table of the *Application Details* panel, choose (Edit) for the respective custom check.
4. Update the endpoint and in a performance check, the thresholds.

   > **i Note**
   >
   > You can't update the type and the name of a check.

   Choose *Update*.

Delete a Custom Check

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Choose *Applications* > *HANA XS Applications* in the navigation area of the subaccount.
3. Select an application from the list, and in the custom checks table of the *Application Details* panel, choose (Delete) for the respective custom check.
4. Confirm the delete operation.

   The row for the check is removed from the table.

Related Information

Alerting [page 779]
SAP BTP Cockpit
Custom HTTP Checks [page 772]
Metrics REST API for SAP HANA XS [page 776]

1.14.5 Metrics REST API for Database Systems

Use the REST API to get metrics for your database systems that are running in the Neo environment.

Protection

The Metrics REST API is available with the following basic URI: `https://api.{host}/monitoring/v2`.

The REST API is protected with OAuth 2.0 client credentials. Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149]. For more information about the format of the REST API, see Metrics API.

> **Note**
>
> While you're creating the API client on the *Platform API* tab, select the *Monitoring Service* API with the *Read Monitoring Data* scope.

Overview

Request the states or the metric details of your database systems by using the `GET` REST API calls.

> ### Example
>
> Use the following request to receive all the metrics for a database system located in the Europe (Rot/Germany) region (with `hana.ondemand.com` host):
>
> `https://api.hana.ondemand.com/monitoring/v2/accounts/<subaccount_technical_name>/dbsystem/<database_system>/metrics`

> ### Example
>
> Use the following request to receive the state of a database system located in the US East (Ashburn) region (with `us1.hana.ondemand.com` host):
>
> `https://api.us1.hana.ondemand.com/monitoring/v2/accounts/<subaccount_technical_name>/dbsystem/<database_system>/state`

1.14.6 Metrics REST API for SAP HANA XS

Use the REST API to get metrics for your SAP HANA XS applications and instances that are running in the Neo environment.

Protection

The Metrics REST API is available with the following basic URI: `https://api.{host}/monitoring/v2`.

The REST API is protected with OAuth 2.0 client credentials. Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149]. For more information about the format of the REST API, see Metrics API.

> ### Note
>
> While you're creating the API client on the *Platform API* tab, select the *Monitoring Service* API with the *Read Monitoring Data* scope.

Overview

Request the states or the metric details of your XS instances by using the `GET` REST API calls. Furthermore, request the states of your XS applications or the checks configured for them.

> ### Example
>
> Use the following request to receive all the checks for an XS application located in the Europe (Rot/Germany) region (with `hana.ondemand.com` host):
>
> `https://api.hana.ondemand.com/monitoring/v2/accounts/<subaccount_technical_name>/instances/<XS_instance_name>/apps/<XS_app_name>/metrics`
>
> You can test this request with the data from your subaccount in the SAP BTP cockpit. On the *HANA XS Applications* page, you can retrieve the names of the application and the instance from the table.

> ### Example
>
> Use the following request to receive the state for an XS application located in the US East (Ashburn) region (with `us1.hana.ondemand.com` host):
>
> `https://api.us1.hana.ondemand.com/monitoring/v2/accounts/<subaccount_technical_name>/instances/<XS_instance_name>/apps/<XS_app_name>/state`

> ### Example
>
> Use the following request to receive all the metrics for an XS instance located in the US West (Chandler) region (with `us2.hana.ondemand.com` host):
>
> `https://api.us2.hana.ondemand.com/monitoring/v2/accounts/<subaccount_technical_name>/instances/<XS_instance_name>/metrics`

> ### Example
>
> Use the following request to receive the state for an XS instance located in the Australia (Sydney) region (with `ap1.hana.ondemand.com` host):
>
> `https://api.ap1.hana.ondemand.com/monitoring/v2/accounts/<subaccount_technical_name>/instances/<XS_instance_name>/state`

1.14.7 Checks REST API for SAP HANA XS Applications

Use the Checks API to retrieve, set, update, or delete custom checks for HTML5 applications in the Neo environment.

Protection

This REST API is protected with OAuth 2.0 client credentials and is used for Java, SAP HANA XS, and HTML5 applications in the Neo environment. Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149]. For more information about the format of the REST APIs, see Checks API.

> ### Note
>
> While you're creating the API client on the *Platform API* tab, select the *Monitoring Service* API with the *Manage Monitoring Configuration* scope.

Overview

You can use this API to configure custom checks for your SAP HANA XS applications. Furthermore, you manage a particular check by specifying the check's type and name. For this request, you can use the following types:

- `custom-http-performance-check`
  This is the default type. With this type, the system uses the configured thresholds to define the metric state. See Custom HTTP Checks [page 772].
- `custom-http-status-check`
  This type allows the system to use the returned HTTP code to define the metric state. See Custom HTTP Checks [page 772].

> ### Example
>
> Setting a Custom Performance Check
>
> Use the following POST request to set the custom check `Bytes read` with the *performance* type:
>
> URI: `https://api.hana.ondemand.com/monitoring/checks/v1/accounts/<subaccount_technical_name>/instances/<XS_instance_name>/apps/<XS_app_name>/types/custom-http-performance-check`
>
> Body:
>
> ```
> [
>   {
>     "checkName": "Bytes read",
>     "url": "https://someappurl.com/metrics/bytesRead.xsjs",
>     "warning": "100",
>     "critical": "200",
>     "unit": "Bytes"
>   }
> ]
> ```

> ### Example
>
> Retrieving a Custom Check
>
> Use the following GET request to retrieve the custom check `Bytes read`:
>
> URI: `https://api.hana.ondemand.com/monitoring/checks/v1/accounts/<subaccount_technical_name>/instances/<XS_instance_name>/apps/<XS_app_name>/types/custom-http-performance-check/checks/Bytes read`

Related Information

Custom HTTP Checks [page 772]
Regions and Hosts Available for the Neo Environment [page 17]
Checks REST API for Java Applications [page 751]
Checks REST API for HTML5 Applications [page 759]

1.15 Alerting

Configure a channel to receive alert notifications when your applications and database systems are in a problematic state or have recovered from such a state.

Overview

You can configure alerting on your custom checks for Java, HTML5, and HANA XS applications, as well as for database systems. There are the following types of notifications:

Notification	Description
CRITICAL	Your application or database system experiences some issues, or the check's metric is above the CRITICAL threshold.
WARNING	The check's metric is above the WARNING threshold but not above the CRITICAL threshold.
OK	The check's metric has recovered from CRITICAL or WARNING state.

The current monitoring tools allow you to configure the following channels for alerts: webhook and email.

- Alert Webhook
  Use the alert webhook channel to configure an application URL that receives the alerts from the SAP Monitoring service. You can configure this channel only with the Alerting Channels REST API.
- Alert Email
  Use the alert email channel to configure alert recipients that receive the alerts by email. You can configure this channel with the Alerting Channels REST API or the Neo console client.

Types of Checks for Alerting

- Default metrics
  The SAP Monitoring service triggers alerts for any changes to the states of the default metrics of a Java application or a database system. For more information, see View Monitoring Metrics of a Java Application [page 730] and View Monitoring Metrics of a Database System [page 1009].

- Availability Checks
  The availability checks are available for Java and SAP HANA XS applications as the alert is triggered if the application isn't in state OK for 2 consecutive checks. For more information, see Availability Checks [page 732].
- JMX Checks
  JMX checks allow alerting on any metric that is based on JMX MBean attribute. The JMX checks are available for Java applications as the alert is triggered after 2 consecutive failures of a JMX check. For more information, see JMX Checks [page 738].
- HTML5 Custom Checks
  The HTML5 custom checks allow you to monitor the heath state of your HTML5 application as the alert is triggered by any change of the check's state. For more information, see Monitoring HTML5 Applications [page 753].
- HANA XS Custom Checks
  The HANA XS custom checks are custom HTTP checks based on a configured URL endpoint as the alert is triggered by any change of the check's state. For more information, see Custom HTTP Checks [page 772].
- Database system checks
  The SAP Monitoring service triggers alerts for any changes to the states of the configured checks for database systems.

Related Information

Alert Webhook [page 780]
Alert Email [page 783]
Alerting Channels REST API
Console Client for the Neo Environment [page 1348]

1.15.1 Alert Webhook

You can use Alerting Channels REST API to configure channels to receive alert notifications. One channel for alert notifications is the alert webhook, which requires you to configure an application URL that receives the alerts from the alerting service of SAP BTP, Neo environment.

Protection

The REST API is protected with OAuth 2.0 client credentials. Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149]. For more information about the format of the REST APIs, see Alerting Channels REST API.

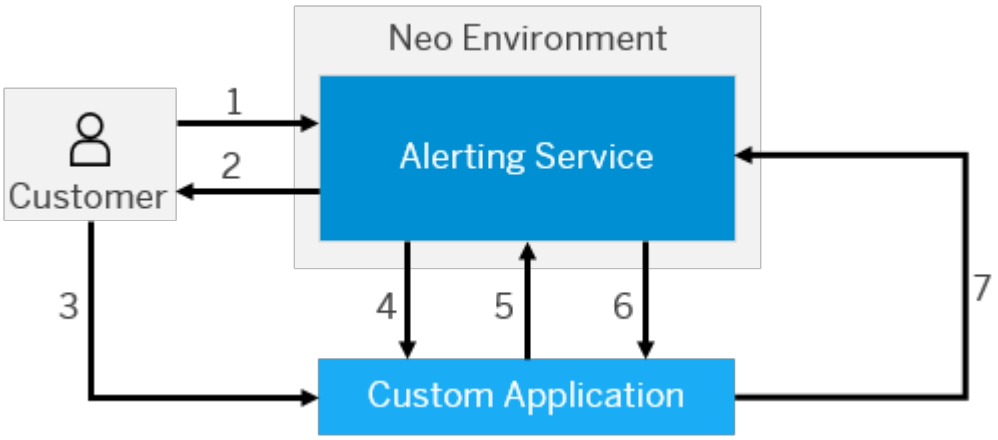
> **i Note**
>
> When you create the API client from the *Platform API* tab in the SAP BTP cockpit, select the *Monitoring Service* API with the *Manage Alerting Channels* scope.

Overview

The alerting service uses webhooks to send alerts to a configured URL. Accordingly, you must implement a custom application and provide a protected URL for such alerts. Your application must also store a verification token that the alerting service requests.

You can use Alerting Channels REST API at the subaccount or application level.

Process Flow



1. The customer sets the application URL for the alerts using a POST REST call to the alerting service. The following is included in the body of the POST request:

   Sample Code

   ```
   {
     "url": "https://customURL.com/alias",
     "authentication": {
                          "authenticationType": "oauth2",
                          "client": "asdfjasdljfasdkjf",
                          "secret": "w7d65as4d1as0das7",
                          "oAuthServerUrl": "https://customServerURL.com/o/
   oauth2/auth"
                        }
   }
   ```

   In this sample, the custom application URL is protected with OAuth 2.0. However, you can also use basic authentication.

   Sample Code

   ```
   "authentication": {
                          "authenticationType": "basic",
                          "user": "yourUser",
                          "password": "123456"
                        }
   ```

The `application` parameter in the JSON is optional and isn't specified for the REST API call at the subaccount level.

2. The alerting service sends a verification token in the response.
3. The customer stores the verification token in the custom application.
4. When the alerting service receives an alert, it sends a GET request to the custom application URL for the verification token.
5. The custom application responds to the GET request.
   The custom application responds with status *200 OK* by sending the verification token as plain text.
6. The alerting service verifies the token, and then sends the alert to the custom application URL using a POST REST call.
   The alert is sent in the following JSON format.

Sample Code

```
{
  "type": "PROBLEM",
  "metric": "My metric name",
  "state": "WARNING",
  "date": "Fri April 03 12:23:05 UTC 2020",
  "output": "JMX WARNING - Value = 2",
  "resourceType": "APPLICATION",
  "resource": {
    "application": "myapp",
    "processId": "9b255a3770c332615f911c09b503eceaf9c1d0b8",
    "account": "mysubaccount"
  }
}
```

The `resourceType` value **APPLICATION**, **DATABASE**, **XS_APPLICATION**, or **HTML5_APPLICATION** defines if the alert is respectively for a Java application, database, SAP HANA XS application, or HTML5 application.

When the alert is for a default or custom metric of a Java application (`"resourceType": "APPLICATION"`), the JSON also includes the process ID. The alerts for the other resource types don't include the process ID.

Sample Code

```
{
  "type": "PROBLEM",
  "metric": "Availability Check",
  "state": "CRITICAL",
  "date": "Thu Sep 20 12:08:20 UTC 2018",
  "output": "CRITICAL: HTTP Status Code: 503 Response size: 222B Response
time: 0.019s",
  "resourceType": "APPLICATION",
  "resource": {
    "application": "myapp",
    "account": "mysubaccount"
  }
}
```

⚠ Caution

The alert for an availability check doesn't include the process ID because of the specifics of this metric.

7. The custom application responds with status code *201 Created* when the operation is successful.

Related Information

Receiving Alerts in the Corporate On-Premise System
Availability Checks [page 732]

1.15.2 Alert Email

Configure alert recipients to receive email notifications for problems with your applications and database systems.

You can use the Neo console client or the Alerting Channels REST API to set the alert recipients for email notifications. To create alert recipients, you use the `set-alert-recipients` console command or the `POST` method of the REST API. The `set-alert-recipients` command allows you to set multiple alert email recipients by specifying the recipient emails separated by comma in a list. However, to configure multiple recipients through the REST API, you have to perform multiple requests as only one recipient is set per a REST API call.

Example: Console Command for Setting Multiple Recipients

```
neo set-alert-recipients -a mysubaccount -b demo -u p1234567 -e
recipient_1@example.com,recipient_2@example.com,...,recipient_n@example.com --
host hana.ondemand.com
```

Example: Body of `POST` Method for the REST API Call

```
{
  "email": "recipient@example.com"
}
```

Related Information

Alert Recipients Commands
list-alert-recipients [page 1479]
set-alert-recipients [page 1538]
clear-alert-recipients [page 1373]

Alerting Channels REST API

Console Client for the Neo Environment [page 1348]

1.16 Auditing and Logging Information

Here you can find a list of the security events that are logged by the SAP Monitoring service.

Security Events Written in Audit Logs

Event grouping	What events are logged	How to identify related log events
Channel-related events **channels**	Create a new channel.	**aboutToCreate** - Creation of channel started **create** - Creation of channel completed `ConfigurationChangeAuditMessage`
	Delete an existing channel.	**aboutToDelete** - Deletion of channel started **delete** - Deletion of channel completed `ConfigurationChangeAuditMessage`
Custom-check-related events **monitoring-customchecks**	Create a custom check.	**aboutToCreate** - Creation of custom check started **create** - Creation of custom check completed `ConfigurationChangeAuditMessage`
	Delete a custom check.	**aboutToDelete** - Deletion of custom check started **delete** - Deletion of custom check completed `ConfigurationChangeAuditMessage`

Event grouping	What events are logged	How to identify related log events
Alert-recipient-related events **alert-recipients**	Configure alert recipients.	**aboutToCreate** - Creation of alert recipients started **create** - Creation of alert recipients completed `ConfigurationChangeAuditMessage`
	Remove alert recipients.	**aboutToDelete** - Removal of alert recipients started **delete** - Removal of alert recipients completed `ConfigurationChangeAuditMessage`
JMX/availability-check-related events **monitoring-configuration-service**	Create a JMX or an availability check.	**create** - Creation of a JMX or an availability check completed `ConfigurationChangeAuditMessage`
	Delete a JMX or an availability check.	**delete** - Deletion of a JMX or an availability check completed `ConfigurationChangeAuditMessage`

Related Information

Audit Logging in the Neo Environment

1.17 Monitoring and Troubleshooting

This section is to help you solve some of the most common issues in SAP Monitoring service.

Getting Support

Context

If you encounter an issue with this service, we recommend following the procedure:

Procedure

1. Check the platform status.

   Check the availability of the platform at SAP Trust Center.

   For more information about platform availability, updates and notifications, see Platform Updates and Notifications in the Neo Environment.
2. Check the troubleshooting scenarios available in the Guided Answers tool.

   See Guided Answers for SAP Monitoring service.
3. Check other resources, and if you don't find a solution, contact SAP Support.

   See Getting Support, Neo Environment [page 1859].

1.18 Remote Data Sync Service

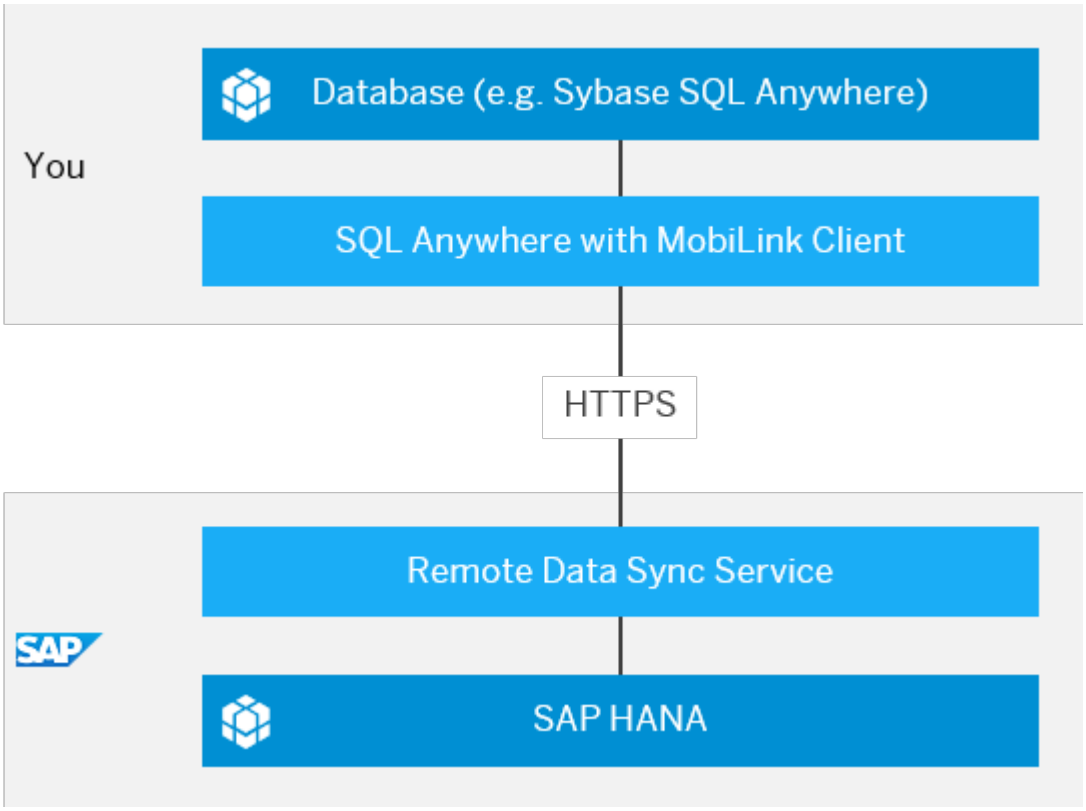
Context

The Remote Data Sync service provides bi-directional synchronization of complex structured data between many remote databases at the edge and SAP BTP databases at the center. The service is based on SAP SQL Anywhere and its MobiLink technology.

- Using Remote Data Sync you can create occasionally-connected applications at the edge. These include applications that are not suitable or economical to have a permanent connection, or applications that must continue to operate in the face of unexpected network failures.
- Also, you can create applications that use a local database and synchronize with the cloud when a connection is available.

- Remote Data Sync allows you to create remote applications that store and share large amounts of data between the application and the cloud. This can significantly reduce latency for data-rich applications and provide a more responsive user experience for remote applications.

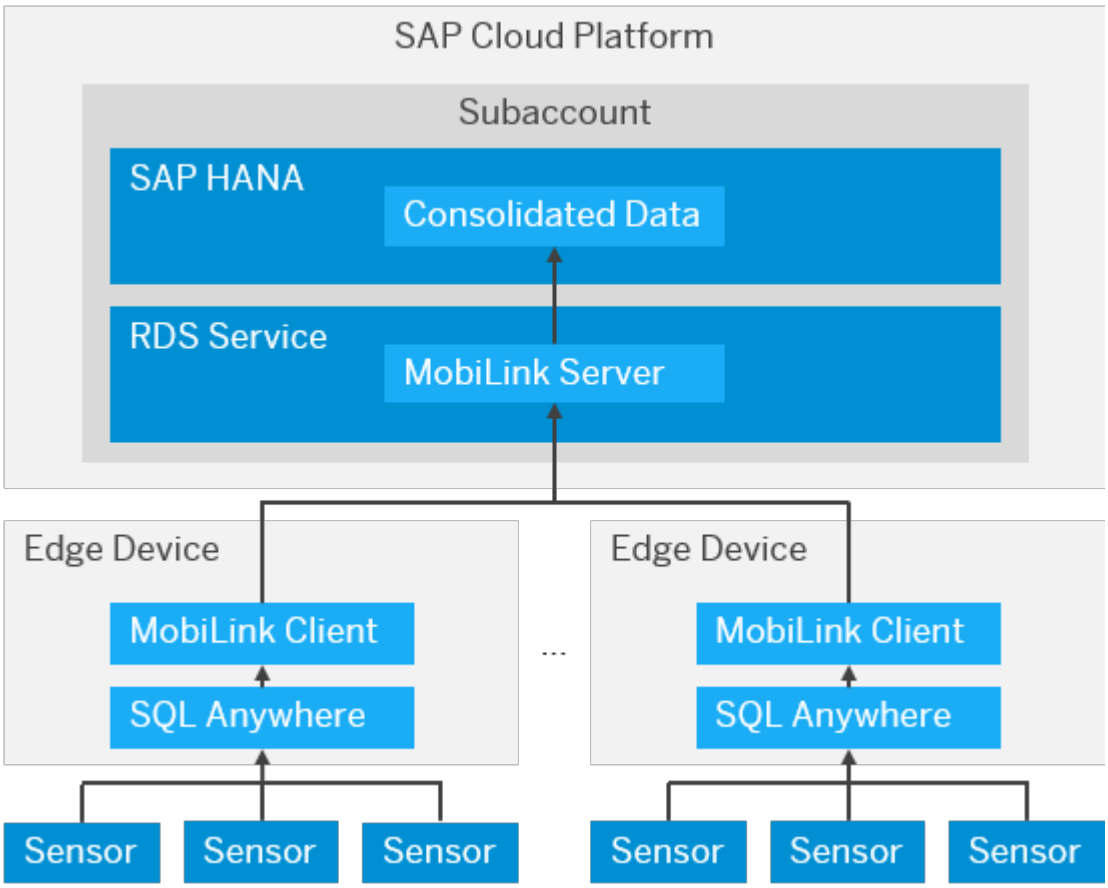
A single cloud database may have hundreds of thousands of data collection and action endpoints that operate in the real world over sometimes unreliable networks. Remote Data Sync provides a way to connect all of these remote applications and to synchronize all databases at the edge into a single cloud database.



The figure below illustrates a typical IoT scenario using the Remote Data Sync service: Sensors or smart meters create data that is sent and stored decentrally in small embedded databases, such as **SQL Anywhere** or **SQL Anywhere UltraLite**. To get a consolidated view on the data of all remote locations, this data is synchronized in the following:

- SAP HANA database on the cloud via SQL Anywhere MobiLink clients, running on the edge devices;
- SQL Anywhere MobiLink servers, which are provided in the cloud by the Remote Data Sync service.

New insights can be later gained by analytics and data mining on the consolidated data in the cloud.



Process Flow

1. Get [page 789] your licenses.
2. Provision [page 790] a MobiLink server in your subaccount which allows you to use the Remote Data Sync service.
3. Develop [page 792] a client-initiated synchronization.
4. Access [page 800] the MobiLink logs during development.
5. Protect [page 802] your MobiLink server.
6. Connect [page 803] the SQL Anywhere tools to the MobiLink server.
7. Configure [page 806] an availability monitor for your MobiLink server.

Sizing

Before you start working with the service, check its sizing requirements and choose the optimal hardware features for fluent run of your applications. For more information, see Performance and Scalability of the MobiLink Server [page 807].

Related Information

SAP SQL Anywhere 16.0 Documentation

1.18.1 Get Access to the Remote Data Sync Service

Prerequisites

- You have an account in a productive SAP BTP landscape (for example, **`hana.ondemand.com`**, **`us1.hana.ondemand.com`**, **`ap1.hana.ondemand.com`**, **`eu2.hana.ondemand.com`**).
- Your SAP BTP account has an SAP HANA instance associated to it. The Remote Data Sync service is currently only supported with SAP HANA database as target database in the cloud.
- On the edge side, you need to install **`SAP SQL Anywhere Remote Database Client`** version 16. You can get a free Developer Edition. See also the existing production packages: Overview

Context

The procedure below helps you to make the Remote Data Sync service available in your SAP BTP account. As the service is not available for your SAP BTP account by default, you need to first fulfill the prerequisites above. After that follow the procedure described below to request the Remote Data Sync service for your account.

> **i Note**
>
> Before you start working with the service, check its sizing requirements and choose the optimal hardware features for fluent run of your applications. For more information, see Performance and Scalability of the MobiLink Server [page 807].

To get access to the Remote Data Sync service, you need to extend your standard SAP BTP license with an a-la-carte license for Remote Data Sync in one of two flavors:

1. Remote Data Sync, Standard: MobiLink server on 2 Cores / 4GB RAM (Price list material number: 8003943)
2. Remote Data Sync, Premium: MobiLink sever on 4 Cores / 8 GB RAM (Price list material number: 8003944)

Next Steps

Provide a MobiLink Server in Your Subaccount [page 790]

1.18.2 Provide a MobiLink Server in Your Subaccount

Prerequisites

- You have received the needed licenses and have enabled the Remote Data Sync service for your subaccount. For more information, see Get Access to the Remote Data Sync Service [page 789].
- You have installed and configured the console client. For more information, see Using the Console Client [page 1348].

Context

To use the Remote Data Sync service, a MobiLink server must be started and bound to the SAP HANA database of your subaccount. This can be done by the following steps (they are described in detail in the procedure below):

1. Deploy the MobiLink server on a compute unit of your subaccount using the console client.
2. Bind the MobiLink server to your SAP HANA database to connect the MobiLink server to the database.
3. Start the MobiLink server within the console client.

> **i Note**
>
> To provision a MobiLink server in your subaccount, you need a free compute unit of your quota. The Remote Data Sync service license includes an additional compute unit for the MobiLink server.

Procedure

1. Deploy the MobiLink server on a compute unit of your subaccount using the `deploy` command. You can configure the MobiLink server to be started with customized server options (see MobiLink Server Options). You can do this either during deployment using the `--ev` parameter, or later on using the `set-application-property` command. You can also specify the compute unit by using the `--size` parameter of the `deploy` command.

   ```
   neo.bat deploy -h <region_host> -a <subaccount> -b
   <mobilink_service_instance> -u <user_name> -s EMPTY_SITE --runtime mobilink
   ```

   - Exemplary MobiLink options configuration during development and starting MobiLink server on a premium compute unit:

     ```
     neo.bat deploy -h hana.ondemand.com -a mysubaccount -b mymlinstance -u
     p1234567890 -s EMPTY_SITE --runtime mobilink --ev ML_ARGS="-zf -v" --size
     prem
     ```

   - Exemplary MobiLink options configuration after deployment.

NOTE: You need to first have a MobiLink server deployed before you can set application properties via the command below. Also, if your MobiLink server has been already started, you'll need to restart the server in order for the changes to take effect. All previously set environment variables will be overridden.

```
neo.bat set-application-property -h hana.ondemand.com -a mysubaccount -b
mymlinstance -u p1234567890 --ev ML_ARGS="-zf -v"
```

2. Bind the MobiLink server to your SAP HANA database. This is needed to connect the MobiLink server to the database.

> **i Note**
>
> **Prerequisite**: You have created an SAP HANA database user dedicated to the MobiLink server instance. For more information, see Creating Database Users [page 1003].
>
> **Hint**: In case your SAP HANA instance is configured to create database users with a temporary password (the user is forced to reset it on first logon), you need to do it before creating the binding.

```
neo.bat bind-hana-dbms -h <region_host> -a <subaccount> -b
<mobilink_service_instance> -u <user_name> -i <hana_instance_name> --db-user
<db_user_name> --db-password <db_user_password>
```

3. Start your MobiLink server:

```
neo.bat start -h <region_host> -a <subaccount> -b <mobilink_service_instance>
-u <user_name>
```

4. Test the state of your MobiLink server.
   - If the application VM is in `Started` state, your server is up and running.
   - If the application VM remains in `App Server Timeout` state and does not manage to start, check the logs of the MobiLink server. See Access MobiLink Server Logs [page 800].

> **i Note**
>
> In case you find the log message below, your binding step is missed or unsuccessfully executed:
>
> ```
> Persistence binding is missing! Please check your binding configuration.
> In case of further issues, contact support team.
> ```

5. You can stop or undeploy your MobiLink server. For more information, see stop [page 1562] or undeploy [page 1574].

Next Steps

Develop Client-Initiated Synchronization [page 792]

1.18.3 Develop Client-Initiated Synchronization

Prerequisites

- An SQL Anywhere version 16 installation is available on the client side. For more information, see Get Access to the Remote Data Sync Service [page 789].
- A MobiLink server is running in your account. For more information, see Provide a MobiLink Server in Your Subaccount [page 790].

Context

This page provides a simple example that demonstrates how to synchronize data from a remote SQL Anywhere database into the SAP HANA database, using the Remote Data Sync service and the underlying SQL Anywhere MobiLink technology. For more information on MobiLink synchronizations, see Quick start to MobiLink (Synchronization).

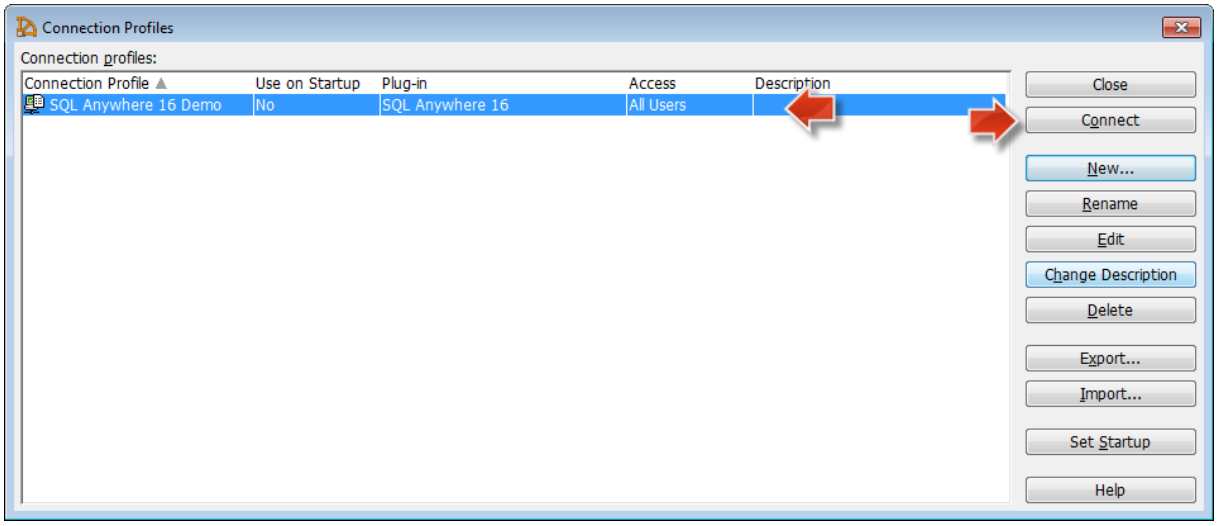
> → Tip
>
> The SQL Anywhere database running on the client side is called **remote database**. The central SAP HANA database running on SAP BTP is called **consolidated database**.

Procedure

1. Connect to a local database

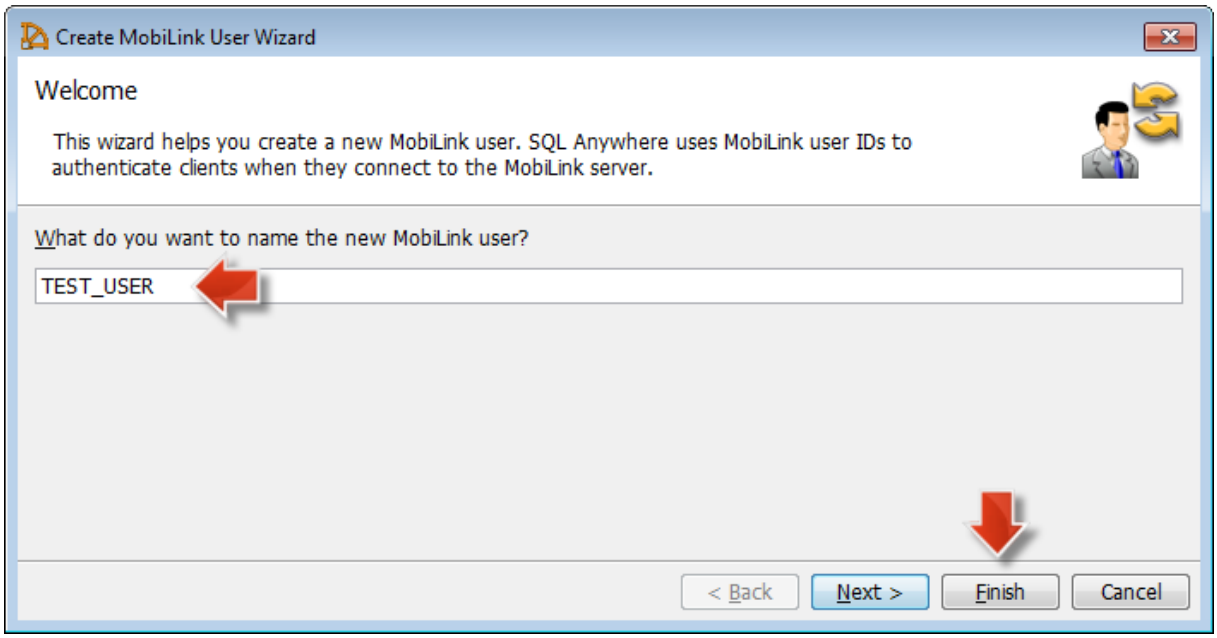
This demo database will be used as a source for the synchronization.

1. Start `Sybase Central`.
2. From the top-level menu, choose *Connections* > *Connection Profiles* > *SQL Anywhere 16 Demo* > profile.
3. Choose *Connect*.



2. Create a new user

1. In `Sybase Central`, double-click *MobiLink Users*.
2. From the context menu, choose *New* > *MobiLink User*.
3. Enter a user name, for example **`TEST_USER`**, and choose *Finish*.
4. Choose the *Back* button in the toolbar menu to get back to the root task level.

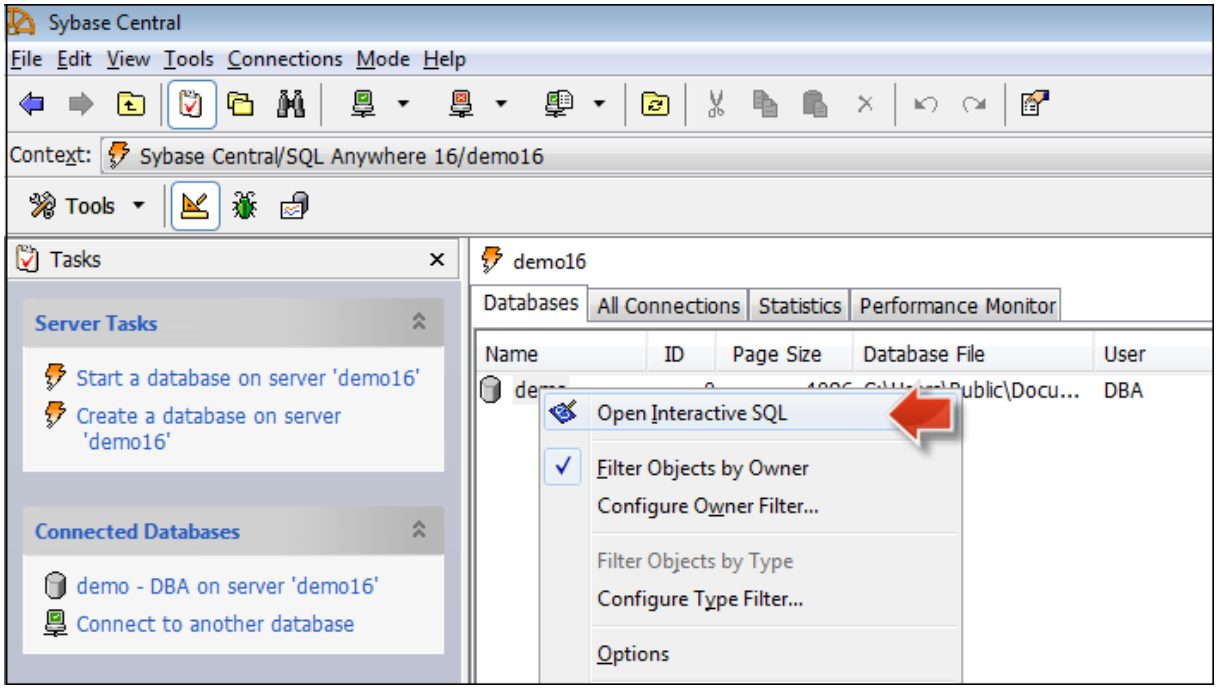


3. Create a new table in the remote database

1. In `Sybase Central`, goto *Database Design Tasks* .
2. Open the context menu above the tabs and choose *Open Interactive SQL*.
3. Execute the following script:

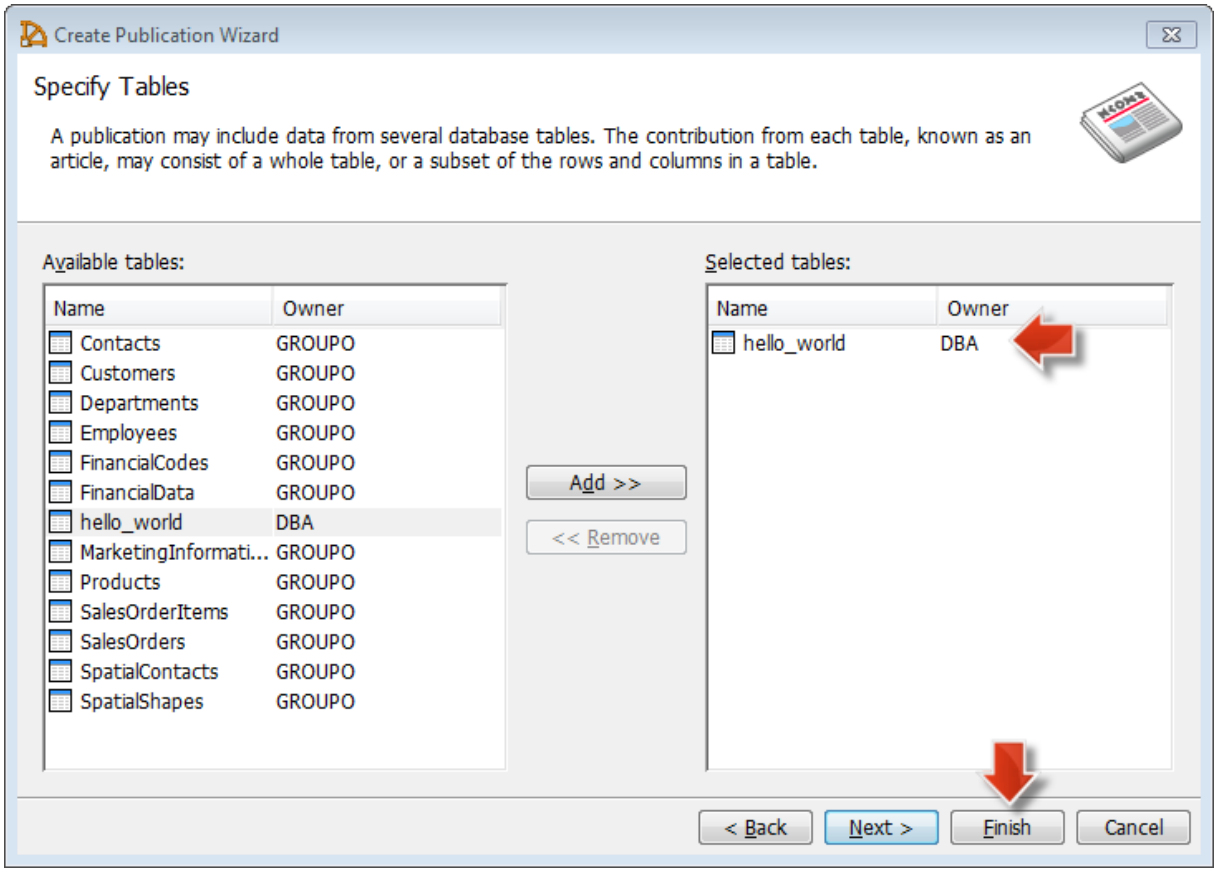
Sample Code

```
CREATE TABLE hello_world (
    pkey BIGINT NOT NULL,
    first_name VARCHAR ( 10 ) DEFAULT '' NOT NULL,
    last_name VARCHAR ( 10 ) DEFAULT '' NOT NULL,
    PRIMARY KEY ( pkey )
);
```



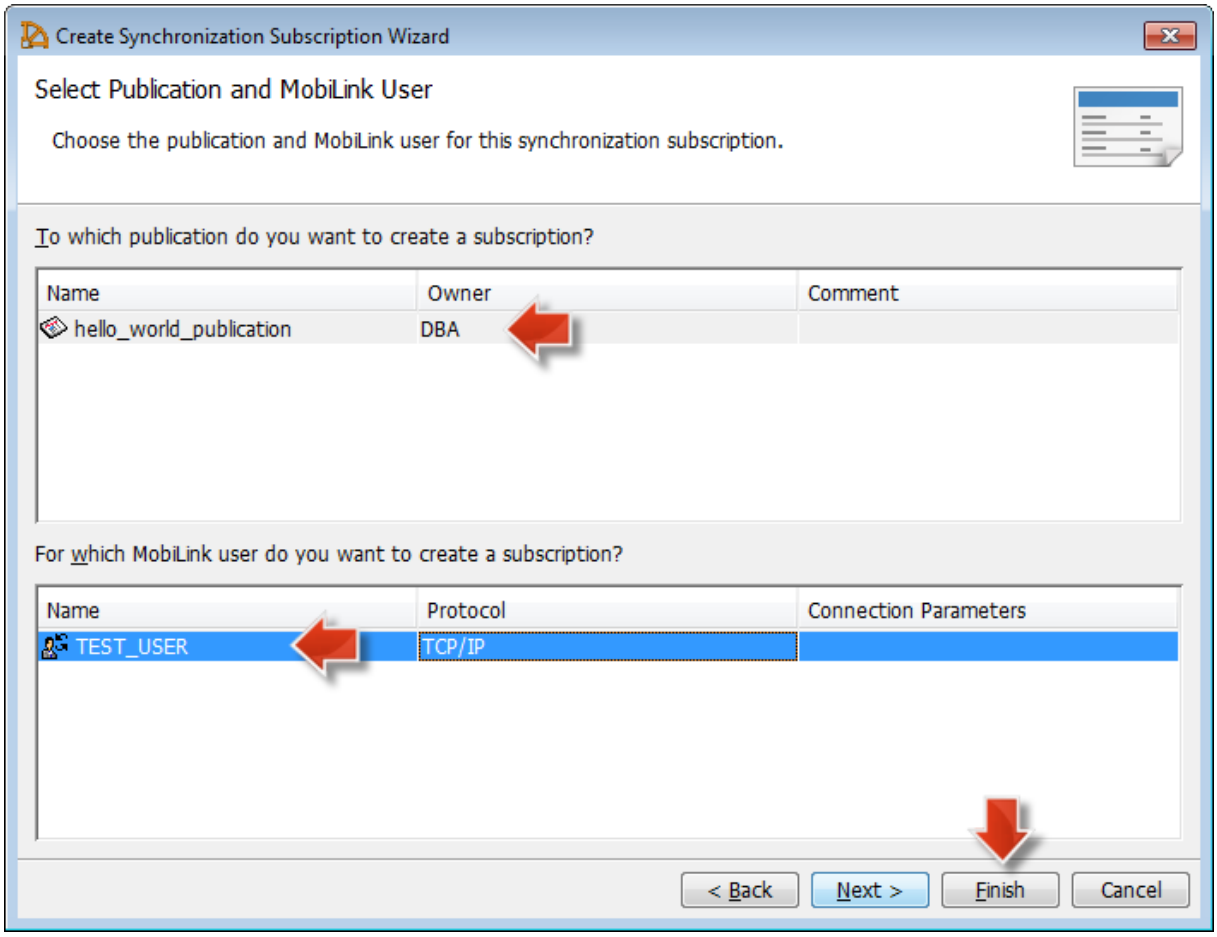
4. Create a publication

1. In `Sybase Central`, double-click *Publications*.
2. From the context menu, select *New* > *Publication*.
3. Enter **`hello_world_publication`** as the name and choose *Next*.
4. Leave the publication type *Log scan* and choose *Next*.
5. Select the *hello_world* table and choose *Finish*.
6. Choose the *Back* button in the toolbar menu to get back to the root task level.



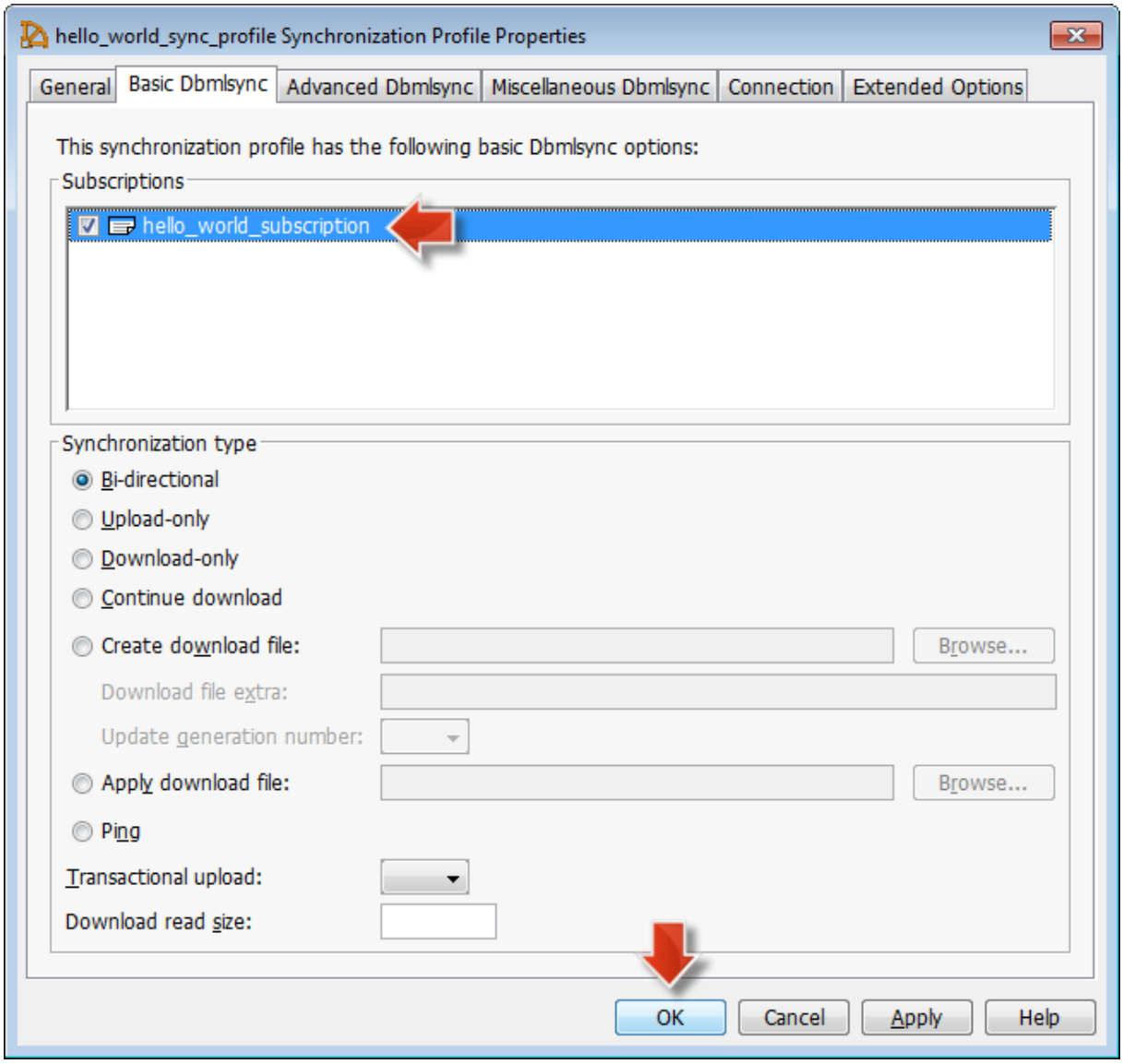
5. Create a synchronization subscription

1. In `Sybase Central`, double-click *Synchronization Subscriptions*.
2. From the context menu, select *New* > *Subscription*.
3. Enter **`hello_world_subscription`** as the name and choose *v1* as the script version.
4. Select *hello_world_publication* and *TEST_USER*, and choose *Finish*.
5. Choose the *Back* button in the toolbar menu to get back to the root task level.



6. Create a synchronization profile

1. In `Sybase Central`, double-click *Synchronization Profiles*.
2. From the context menu, select *New* > *Synchronization Profile*.
3. Enter **`hello_world_sync_profile`** as the name.
4. Skip the comment wizard and choose *Finish*.
5. In the new *Synchronization Profile Properties* view, select *hello_world_subscription* and choose *OK*.
6. Set the script version number for Synchronization Profile in the *Extended Options* tab and choose *OK*.
7. Choose the *Back* button in the toolbar menu to get back to the root task level.

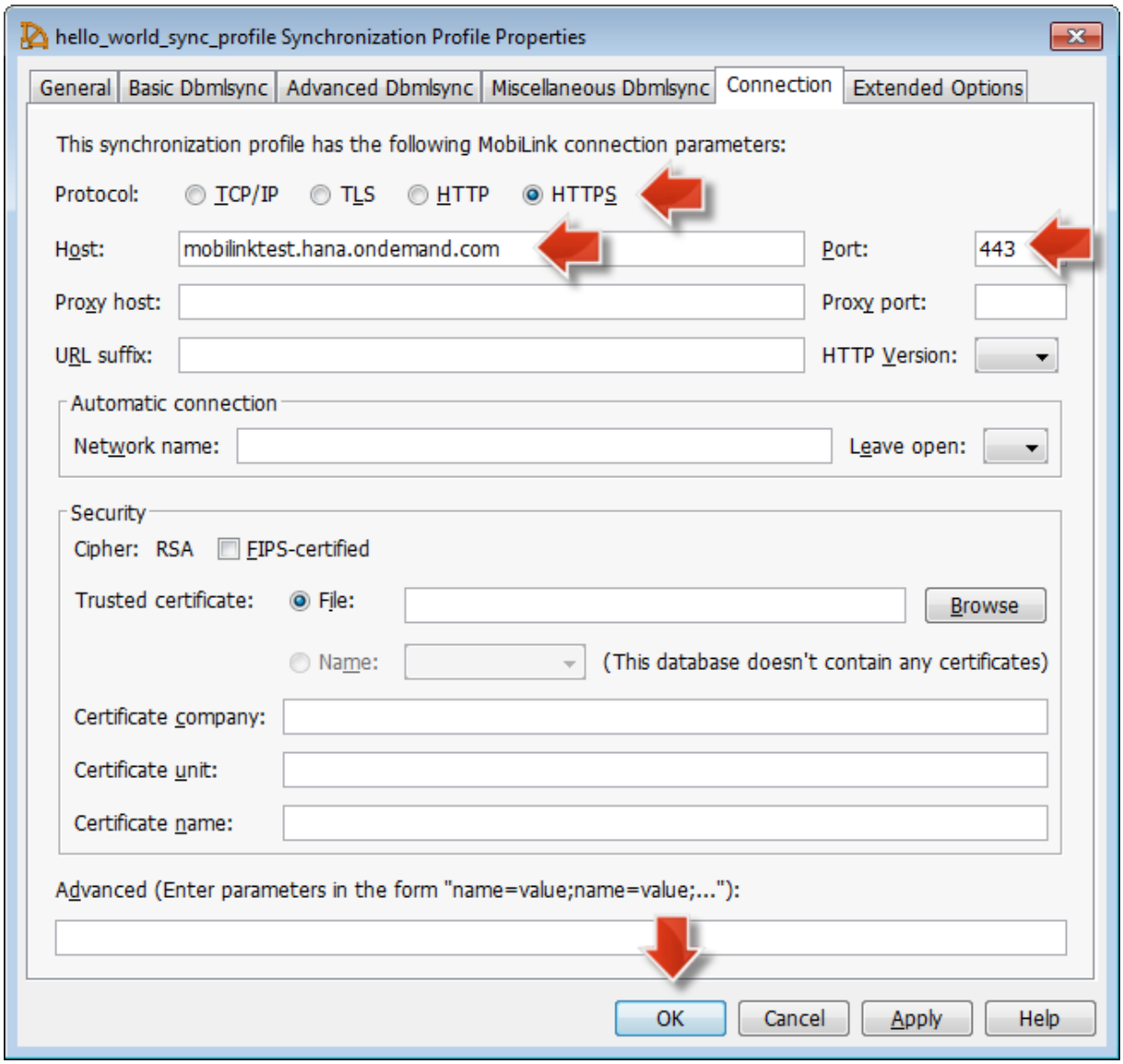


7. Connect the MobiLink client to the MobiLink server

1. In Sybase Central, double-click Synchronization Profiles.
2. Select hello_world_sync_profile, and from its context menu, choose Properties.
3. Go to the Connection tab.
4. Fill in the connection details of the MobiLink server as described below:
   - *Protocol*: `HTTPS`
   - *Host*: URL to the MobiLink server. To see this URL, open the cockpit and navigate to *Java Applications*. From the list of applications, select the MobiLink server (NOTE: as of today, it appears as a Java application), and then copy the *Application URL* for the application shown in the *Application Details* view.
   - *Port*: `443`

- *Proxy host*: define your local proxy host, if available
- *Proxy port*: define your local proxy port, if available

5. Choose *OK*.
6. Choose the *Back* button in the toolbar menu to get back to the root task level.



8. Insert sample data in the hello_world table

1. In `Sybase Central`, select the *demo* database.
2. From its context menu, choose *Open Interactive SQL*.
3. Execute the following script:

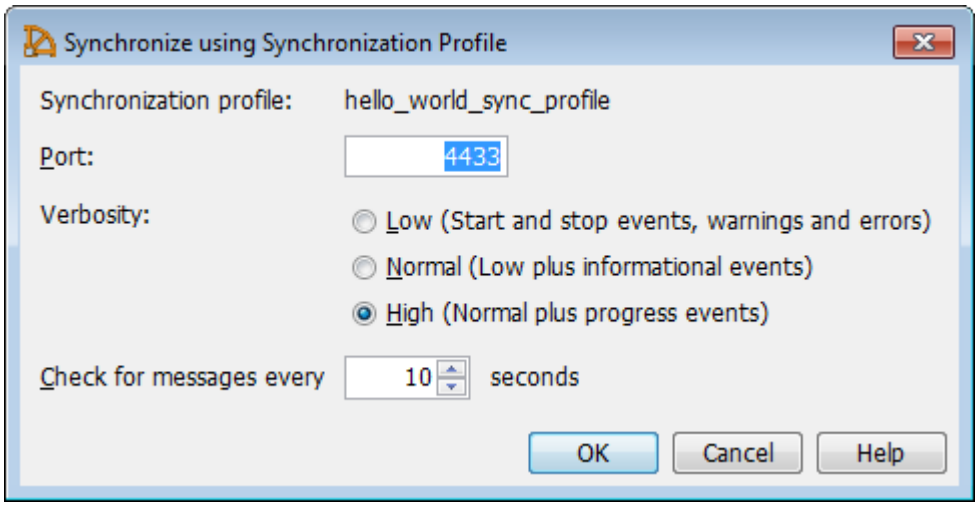
Sample Code

```
insert into hello_world (pkey, first_name, last_name) values (50, 'John',
'Miller');'
insert into hello_world (pkey, first_name, last_name) values (52,
'Olivia', 'Snider');
```

4. Choose the *Back* button in the toolbar menu to get back to the root task level.

9. Run a synchronization

1. In `Sybase Central`, double-click *Synchronization Profiles*.
2. Select *hello_world_sync_profile*, and from its context menu, choose *Synchronize* and then *OK*.
3. Check for errors in the synchronization output. The lines will be colored in red if there is a sync failure.
4. Use SAP HANA Studio to validate that the data from the remote database has been synchronized into the consolidated SAP HANA database.
5. Choose the *Back* button in the `Sybase Central` toolbar menu to get back to the root task level.



Next Steps

Access MobiLink Server Logs [page 800]

Audit Logging of MobiLink Synchronizations [page 801]

1.18.4 MobiLink Server Logs

Tasks

- Access MobiLink Server Logs [page 800]
- Audit Logging of MobiLink Synchronizations [page 801]

Related Information

Provide a MobiLink Server in Your Subaccount [page 790]

1.18.4.1 Access MobiLink Server Logs

Context

You can access the MobiLink server logs both in the cockpit and the console client.

Procedure

Accessing Logs in the Cockpit

1. Open the cockpit.
2. In the left navigation, choose *Java Applications*. For the time being, your MobiLink server appears as a Java application.
3. Find the MobiLink server name, which you have specified during MobiLink provisioning, and choose it.
4. In the *Most Recent Logging* section, click the icon to view the logs, or the icon to download them.

Accessing Logs in the Console Client

1. Open the console client, navigating to the `<SDK_installation>/tools` directory.
2. To list all available logs of the MobiLink runtime, execute the `list-logs` command below (exemplary data is given):
   Example:

```
neo list-logs --account mysubaccount --application mymobilinkserver --user
p1234567890 --host hana.ondemand.com
```

3. To download the log file of the MobiLink server, execute the `get-log` command below (exemplary data is given):
   Example:

```
neo get-log --account mysubaccount --application mymobilinkserver --user
p1234567890 --host hana.ondemand.com --directory C:\MyMobiLink\log --file
mobilink_runtime_2015-03-10.log
```

Related Information

Audit Logging of MobiLink Synchronizations [page 801]
list-logs [page 1496]
get-log [page 1448]

1.18.4.2 Audit Logging of MobiLink Synchronizations

This page helps you to achieve end-to-end traceability of all synchronizations done via the Remote Data Sync service of SAP BTP. This way, you can track who made what changes during work on the SAP HANA target database in the cloud.

SAP HANA Auditing Activity

To monitor and record which users performed selected actions on SAP HANA database, you can use the SAP HANA Audit Activity with Database Table as trail target. To use this feature, it must first be activated for your SAP HANA database. This can be done via SAP HANA Studio by a database user with role **HCP_SYSTEM**.

- Use an SAP HANA database table as the trail target makes it possible to query and analyze auditing information quickly. It also provides a secure and tamper-proof storage location.
- Audit entries are only accessible through the public system view **AUDIT_LOG**. Only SELECT operations can be performed on this view by users with the system privilege **`AUDIT OPERATOR`** or **`AUDIT ADMIN`**.

For more information about how to configure audit policy, see SAP HANA Administration Guide and SAP HANA Security Guide.

> **i** Note
>
> These links point to the latest release of *SAP HANA Administration Guide* and *SAP HANA Security Guide*. Refer to the SAP BTP Release Notes to find out which HANA SPS is supported by SAP BTP. Find the list of guides for earlier releases in the *Related Links* section below.

MobiLink Server Log as Audit Log

Additionally to the SAP HANA audit logs, you might want to use the MobiLink server logs to achieve end-to-end traceability.

- We recommend that you set the log level of the MobiLink server to a value that produces logs in granularity useful for end-to-end traceability of the performed synchronization operations. For example, the log level -vtRU. For more information about this log level configuration, see -v parameter documentation.
- To configure the log level, use the deploy command in the console client. For more information, see Provide a MobiLink Server in Your Subaccount [page 790].

> → Remember
>
> SAP BTP retains the MobiLink server log files for only a week. To fulfill the legal requirements regarding retention of audit log files, make sure you download the log files regularly (at least once a week), and keep them for a longer period of time according to your local laws.

Related Information

Access MobiLink Server Logs [page 800]
Guides for earlier releases of SAP HANA

1.18.5 Configure Authentication for a MobiLink Server

Context

This section provides information about security-related operations and configurations you can perform in a Remote Data Sync scenario.

Currently, as part of SAP BTP, the MobiLink servers support only `basic authentication`. For more information, see User Authentication Architecture.

Tasks

- Creating MobiLink Users in a Consolidated Database
- Configuring a MobiLink Client to Use Transport-Layer Security using the dbmlsync -e option.

> **Note**
>
> On SAP BTP, MobiLink clients can only be connected via HTTPS to MobiLink servers in the cloud, which means that plain HTTP connections are not supported.

There are different options how to configure the HTTPS connection, depending on the SQL Anywhere synchronization tool that is used to trigger synchronizations:

- When using SQL Anywhere `dbmlsync` command line tool to trigger client-initiated synchronizations, trusted certificates can be specified using the `trusted_certificates` parameter as described here .
- When using the Sybase Central UI to trigger client-initiated synchronizations, you can specify `Trusted certificates` as described here .

Related Information

MobiLink Users

MobiLink Security

SQL Anywhere Certificate Utilities

MobiLink Client Configuration to use TLS

SQL Anywhere User and Database Security

MobiLink Client/Server Communication Encryption

1.18.6 Connecting SQL Anywhere Tools to MobiLink Servers

Prerequisites

- An SQL Anywhere version 16 installation is available on the client side. For more information, see Get Access to the Remote Data Sync Service [page 789].
- A MobiLink server is running in your account. For more information, see Provide a MobiLink Server in Your Subaccount [page 790].

Context

The page describes how existing tools of SQL Anywhere (`SQL Anywhere Monitor` and `MobiLink Profiler`) can be connected and used with the Remote Data Sync service running on SAP BTP.

Tasks

- Connect an SQL Anywhere Monitor [page 804]
- Connect the MobiLink Profiler [page 805]

Related Information

SQL Anywhere Monitor

MobiLink Profiler

1.18.6.1 Connect an SQL Anywhere Monitor

Context

SQL Anywhere Monitor comes as part of the standard SQL Anywhere installation. You can find it under *Administrative Tools* of SQL Anywhere 16. The tool provides basic information about the health and availability of a SQL Anywhere and MobiLink landscape. It also gives basic performance information and overall synchronization statistics of the MobiLink server.

Procedure

1. To start the SQL Anywhere Monitor tool, open the SQL Anywhere 16 installation and go to *Administrative Tools*.
2. Open the SQL Anywhere Monitor dashboard via URL: **`http://<host_name>:4950`**, where `<host_name>` is the host of the computer where SQL Anywhere Monitor is running.
3. Log in with the default credentials: user= **`admin`** , password= **`admin`** .
4. In the dashboard, go to *Tools* > *Administration* > *Resources* > .
5. Choose *Add*.
6. Select *MobiLink Server* and proceed with the wizard, providing the following details:
   - MobiLink server:
     - As *Host*, specify the fully qualified domain name of the MobiLink server running in your SAP BTP account.
     - As *Port*, specify **`8443`**.
     - As *Connection Type*, specify **`HTTPS`**. Leave the rest unchanged.
   - MobiLink user: provide the credentials of a valid MobiLink user.

- Collection interval: time interval after which SQL Anywhere Monitor contacts the MobiLink server again to fetch data

7. Once the resources have been added, you can start monitoring the MobiLink server and add widgets to show different performance metrics, such as `Sync Metrics`, `MobiLink Server Info`, `Raw Metrics`, and so on.

Next Steps

SQL Anywhere Monitor also allows you to configure e-mail alerts for synchronization problems. For more information, see Alerts.

Related Information

Monitor the Availability of the MobiLink Server [page 806]

1.18.6.2 Connect the MobiLink Profiler

Context

MobiLink Profiler comes as part of the standard SQL Anywhere installation. You can find it under *Administrative Tools* of SQL Anywhere 16. The tool collects statistical data about all synchronizations during a profiling session, and provides performance details of the single synchronizations, down to the detailed level of a MobiLink event. It also provides access to the synchronization logs of the MobiLink server. Therefore, the tool is mostly used to troubleshoot failed synchronizations or performance issues, and during the development phase to further analyze synchronizations, errors, or warnings.

Procedure

1. Start the MobiLink Profiler under *Administrative Tools* of SQL Anywhere 16. The tool is a desktop client and does not run in a Web browser.
2. Open *File* > *Begin Profiling Session* to connect to the MobiLink server of your cloud account.
3. In the *Connect to MobiLink Server* window, provide the appropriate connection details, such as:
   - *Host*: specify the fully qualified domain name of the MobiLink server running in your SAP BTP account.
   - *Port*: **8443**

- *User/Password*: the credentials of a valid MobiLink user.
- *Protocol*: **`HTTPS`**
- Trusted certificate file: Needed in case certificate based authentication is used to connect to the MobiLink server. You need to specify the local path to the certificate file.
- Additional protocol options: Specify additional protocol options, such as `proxy_host` and `proxy_port` if needed for connecting to the MobiLink server.

Next Steps

To learn more about the UI of the MobiLink Profiler, see MobiLink Profiler Interface.

1.18.7 Monitor the Availability of the MobiLink Server

Prerequisites

- An SQL Anywhere version 16 installation is available on the client side. For more information, see Get Access to the Remote Data Sync Service [page 789].
- A MobiLink server is running in your subaccount. For more information, see Provide a MobiLink Server in Your Subaccount [page 790].

Context

This page describes how you can configure an availability check for your MobiLink server and subscribe recipients to receive alert e-mail notifications when your server is down or responds slowly. Furthermore, recommended actions are listed in case of issues.

Procedure

1. Open the console client, navigating to the `<SDK_installation>/tools` directory.
2. To create the availability check, execute the following command (exemplary data).

   Example:

   ```
   neo create-availability-check -a mysubaccount -b mymlinstance -u p1234567890 -
   U /status -C 6 -W 4 -h hana.ondemand.com
   ```

3. To subscribe recipients to notification alerts, execute the following command (exemplary data).

Example:

```
neo set-alert-recipients -a mysubaccount -b mymlinstance -u p1234567890 -e
john.smith@google.com -h hana.ondemand.com
```

> → Tip
>
> To add multiple e-mail addresses, separate them with commas. We recommend that you use distribution lists rather than personal e-mail addresses. Keep in mind that you will remain responsible for handling of personal e-mail addresses with respect to data privacy regulations applicable.

Next Steps

Recommended actions in case of issues:

- Check the logs. In case of synchronization errors, use the `MobiLink Profiler` tool to drill down into the problem for root cause analysis.
- In case of crude server startup parameters, reset the MobiLink server.
- If your MobiLink server hangs, restart it.

Related Information

Configure Availability Checks for Java Applications from the Console Client [page 737]

1.18.8 Performance and Scalability of the MobiLink Server

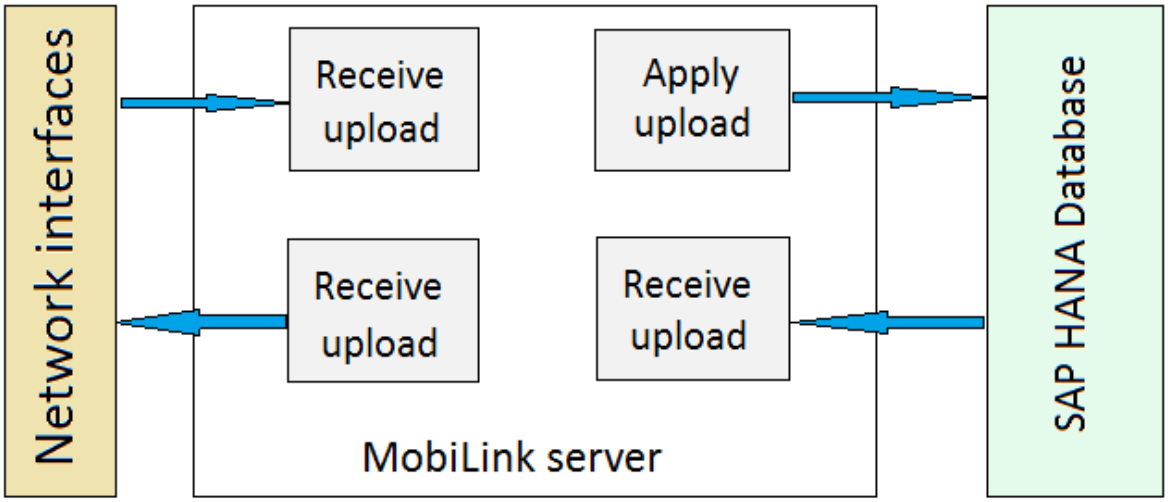
This page provides sizing information for applications using the Remote Data Sync service.

Although the only realistic answers to optimal resource planning are “It depends” and “Testing will show what you need”, this section aims to help you choose the right hardware parameters.

Synchronization Phases

The figure below shows the major phases of a synchronization session. Though not complete, it covers many common use cases.

1. Synchronization is initiated by a remote database client. It uploads any changes made at the remote database to the server.
2. MobiLink applies the changes to the database.
3. MobiLink queries the database and prepares the changes to be sent to the remote database client.
4. MobiLink sends the changes to the remote database client.



Roughly, the MobiLink server uses two thread pools – one for database connections, and one for the network side. These can be controlled by command-line options, although, by default, the Remote Data Sync service dynamically tunes the size of the worker thread pool to accommodate load changes.

Database Capacity

When the Remote Data Sync server applies changes to the consolidated database and prepares changes to be sent to the remote database client, it typically does so by executing SQL statements or stored procedures that are invoked by MobiLink events. For example, to apply an upload MobiLink may execute insert, update, and delete statements for each table being synchronized; to prepare a download MobiLink may execute a query for each table being synchronized.

Database tuning is outside the scope of this document, but the load on the database can be substantial. Think of MobiLink as a concentrator of database load. All the operations that are carried out against the remote database while disconnected, in addition to the requests for updates to be downloaded to the remote database, are executed in two transactions (1 upload, 1 download) against the consolidated database. This can place a heavy load on the database.

Maximum Number of Concurrent Synchronizations

You should know the number of concurrent synchronizations as a starting point, and from there on, calculate back on the required resources. Typically, this number is limited by RAM requirements. To estimate, you need a typical upload and download data volume as a starting point.

A machine with **N** MB of RAM can have **C** clients each with about **V** MB of upload or download data volume, where **C = N/V**.

Following this formula, for large synchronizations (< 20 MB), you can have:

- `Remote Data Sync service, Standard`: C = 4096/20 → C @ 200 simultaneous syncs

- `Remote Data Sync service, Premium`: C = 8192/20 → C @ 400 simultaneous syncs

Maximum Sustainable Throughput

The rate limiting steps of a synchronization are commonly:

- Data transfer, in the case of slow networks
- Database operations, especially in the case of complex schemas and complex synchronization logic.

Remote Data Sync servers are not typically CPU intensive, and typically require less than half the processing that is required by the consolidated database. When selecting the appropriate compute units for MobiLink, memory is more likely to limit the maximum sustainable throughput for a Remote Data Sync server than CPU.

Example:

1. Let's assume the database can process the target load of **`L`** synchronizations per second (and that is a matter for testing).
2. At this throughput, one database thread will come open every **`1/L`** seconds. To keep throughput high, a synchronization request should be ready, with data uploaded and available to pass to the database thread.
3. To keep the database busy, if a synchronization request takes **`t`** seconds to upload (which will depend on network speed and data volume, and which should be determined by testing), then the Remote Data Sync server must be able to hold (**`L`** x **`t`**) client data uploads in memory.
4. The Remote Data Sync server must also be able to download the data to the client to prevent the database threads having to wait for a network connection to download. In the case, this volume is similar to the uploads we end up with: MobiLink should be able to support (**`2`** x **`L`** x **`t`**) simultaneous synchronizations to maintain a throughput of **`L`** synchronizations per second.

> **i** Note
>
> For example, to support a peak sustained throughput of **`50`** synchronizations per second, with a client that takes **`0.5`** seconds to upload and download data, then the Remote Data Sync server should be able to support **`50`** simultaneous synchronizations in RAM to sustain this rate as a peak throughput. Assuming data transfer volumes per client are less than **`80`** MB (which is a very high number for data synchronization), a `Standard` machine would be a good choice to start with.

1.19 Tutorials

Follow the tutorials below to get familiar with the services offered by SAP BTP in the Neo environment.

To learn about the Neo environment	See
How to create a "Hello World" Multi-Target Application	Create a Hello World Multitarget Application [page 1017]

To learn about the Neo environment	See
Connectivity service scenarios	Consume Internet Services (Java Web or Java EE 6 Web Profile) [page 155] Consume Internet Services (Java Web Tomcat 7) [page 163] Consume Backend Systems (Java Web or Java EE 6 Web Profile) [page 170] Consume Backend Systems (Java Web Tomcat 7) [page 181] Invoke ABAP Function Modules in On-Premise ABAP Systems [page 194] Example: Send E-Mails [page 211]
SAP HANA and SAP ASE service scenarios	Tutorial: Adding Application-Managed Persistence with JPA (SDK for Java Web) [page 931] Tutorial: Adding Persistence with JDBC (SDK for Java Web) [page 966] Creating an SAP HANA Database Using the Console Client
Java applications lifecycle management scenarios	Java ALM API Tutorial [page 877]
Virtual machine scenarios	Virtual Machine Tutorials and Guides
Monitoring service scenarios	Implement a Dashboard Application for Monitoring Java Apps Implement a Notification Application for Monitoring Java Apps
How to secure your HTTPS connections	Using the Keystore Service for Client Side HTTPS Connections [page 1791]
How to create an SAP HANA XS application	• Creating an SAP HANA XS Hello World Application Using SAP HANA Studio [page 989] • Creating an SAP HANA XS Hello World Application Using SAP HANA Web-based Development Workbench [page 985]
Multitenancy scenarios	Create an Exemplary Provider Application (Servlet) [page 905] Create an Exemplary Provider Application (JSP) [page 908] Create a Multitenant Connectivity Application [page 911]

More Tutorials

openSAP Course Videos: Introduction to SAP BTP

1.20 Tools

An overview of tools for working in the Neo environment of SAP BTP.

Tool	Description
SAP BTP Cockpit	The SAP BTP cockpit is the web-based administration interface of SAP BTP and provides access to a number of functions for configuring and managing applications, services, and subaccounts. Use the cockpit to manage resources, services, security, monitor application metrics, and perform actions on cloud applications.
Account Administration Using the SAP BTP Command Line Interface (btp CLI) [Feature Set B] [page 1317]	The SAP BTP command line interface (btp CLI) is the command-line tool for convenient account management, such as managing global accounts, directories, and subaccounts.
SAP Web IDE [page 815]	SAP Web IDE is a cloud-based meeting space where multiple application developers can work together from a common Web interface — connecting to the same shared repository with virtually no setup required. SAP Web IDE allows you to prototype, develop, package, deploy, and extend SAPUI5 applications.
Maven Plugin [page 815]	The Maven plugin supports you in using Maven to develop Java applications to the Neo environment. It allows you to conveniently call the console client and its commands from the Maven environment.
Cloud Connector [page 232]	The Cloud Connector serves as the link between on-demand applications in the Neo environment. This is the browser-based and existing on-premise systems. You can control the resources available for the cloud applications in those systems.
SAP BTP SDK for Neo Environment [page 816]	The SAP BTP SDK for the Neo environment contains everything you need to work with the Neo environment , including a local server runtime and a set of command line tools.
SAP BTP SDK for iOS [page 817]	The SAP BTP SDK for iOS is based on the Apple Swift programming language for developing apps in the Xcode IDE and includes well-defined layers (SDK frameworks, components, and platform services) that simplify development of enterprise-ready mobile native iOS apps. The SDK is tightly integrated with SAP Mobile Services for Development and Operations.
SAP JVM Tools in Eclipse for the Neo Environment [page 818]	SAP JVM Tools for Eclipse is an Eclipse plugin that allows you to debug and profile applications which run on SAP JVM.

Tool	Description
Console Client for the Neo Environment [page 1348]	The console client for the Neo environment enables development, deployment and configuration of an application outside the Eclipse IDE as well as continuous integration and automation tasks.

Related Information

Working with Cloud Management Tools Feature Set B in the Neo Environment [page 819]

1.20.1 SAP BTP Cockpit

A web-based administration interface provides access to a number of functions for configuring and managing applications, services, and subaccounts.

Use the cockpit to manage resources, services, security, monitor application metrics, and perform actions on cloud applications.

The cockpit provides an overview of the applications available in the different technologies supported by SAP BTP, and shows other key information about the subaccount. The tiles contain links for direct navigation to the relevant information.

Home Page

The first thing you see on SAP BTP is the home page. You can find key information about the cloud platform and its service offering. You can log on to the cloud cockpit from the home page, or register if you are a new user.

When you log on to the cockpit, you see one or more global accounts. Choose a global account to work in (based on your contract).

Logon

Log on to the cockpit using the relevant URL. The URL depends on the region. The logon URL differs in terms of the physical location where applications, data, or services associated with the account are hosted. For example, use `https://account.hana.ondemand.com/cockpit` to log on to a customer or partner account located in Europe.

For more information, see Regions and Hosts Available for the Neo Environment [page 17].

> **i Note**
>
> We recommend that you log on with your e-mail address.

Accessibility

SAP BTP provides High Contrast Black (HCB) theme support. Switch between the default theme and the high contrast theme by choosing *Your Name* > *Settings* in the header toolbar and selecting a theme. Once you have saved your changes, the cockpit starts with the theme of your choice.

The cockpit icons are designed in high-contrast mode.

Language

To set the language in which the cockpit should be displayed, choose one of the following options from *Your Name* > *Settings* in the header toolbar:

- English
- Japanese
- Chinese
- Korean

Browser Support

For more information, see the Prerequisites and Restrictions.

Notifications

Use *Notifications* to stay informed about different operations and events in the cockpit, for example, to monitor the progress of copying a subaccount.

For more information, see Notifications [page 814].

Get Support

To ask a question or give us feedback, choose one of the following options from *Your Name* > *Settings* in the header toolbar:

- *Technical Assistance* to request product support from SAP.
- *Non-Technical Assistance* for all other questions.
- *Give Feedback* to tell us what you think about using SAP BTP.

Related Information

Account Model [page 10]

1.20.1.1 Notifications

Use *Notifications* to stay informed about different operations and events in the cockpit, for example, to monitor the progress of copying a subaccount.

The *Notification* icon in the header toolbar provides a quick access to the list of notifications and shows the number of available notifications. The icon is visible only if there are currently notifications.

Each notification includes a short statement, a date and time, and the relevant subaccount. A notification informs you about the status of an operation or asks for an action. For example, if copying a subaccount failed, an administrator of the subaccount can assign the corresponding notification to himself and provide a fix. The other members of this subaccount can see that the notification is already assigned to someone else.

You have the following options:

- Dismiss a notification.
- Assign a notification to yourself. It's possible also to unassign yourself from a notification without processing it further.
- Once you have completed the related action, you can set the status to complete. This dismisses the corresponding notification for everyone else.

You can access the full list of notifications (also the ones you have dismissed earlier) by choosing *Notifications* in the navigation area at the region level.

Related Information

SAP BTP Cockpit [page 812]

1.20.2 SAP Web IDE

SAP Web IDE is a fully extensible and customizable experience that accelerates the development life cycle with interactive code editors, integrated developer assistance, and end-to-end application development life cycle support. SAP Web IDE was developed by developers for developers.

SAP Web IDE is a next-generation cloud-based meeting space where multiple application developers can work together from a common Web interface — connecting to the same shared repository, with virtually no setup required. It includes multiple interactive features that allow you to collaborate with your project colleagues and prototype, develop, package, deploy, and extend SAPUI5 applications.

Related Information

SAP Web IDE documentation

1.20.3 Maven Plugin

SAP offers a Maven plugin that supports you in using Maven to develop Java applications for SAP BTP. It allows you to conveniently call the SAP BTP console client and its commands from the Maven environment.

Most commands that are supported by the console client are available as goals in the plugin. To use the plugin, you require a SAP BTP SDK for Neo environment, which can be automatically downloaded with the plugin. Each version of the SDK always has a matching Maven plugin version.

For a list of goals and parameters, usage guide, FAQ, and examples, see:

Maven Plugin Documentation

Related Information

Building Java Web Applications with Maven
Working with the "Neo" Maven Plugin

1.20.4 SAP BTP SDK for Neo Environment

The SAP BTP SDK for Neo environment contains everything you need to work with SAP BTP, including a local server runtime and a set of command line tools.

Prerequisites

You have the SDK installed. See Install the SAP BTP SDK for Neo Environment [page 845].

The location of the SDK is the folder you have chosen when you downloaded and unzipped it.

An overview of the structure and content of the SDK is shown in the table below. The folders and files are located directly below the common root directory in the order given:

Folder/File	Description
`api`	The platform API containing the SAP and third-party API JARs required to compile Web applications for SAP BTP (for more information about the platform API, see the "Supported APIs" section further below).
`javadoc`	Javadoc for the SAP platform APIs (also available as online documentation via the *API Documentation* link in the title bar of the SAP BTP Documentation Center). Javadoc for the third-party APIs is cross-referenced from the online documentation.
`repository`	The P2 repository from which the local server runtime is created.
`samples`	Samples demonstrating how to develop for SAP BTP. The samples can be imported as Eclipse or Maven projects.
`server`	Initially not present, but created once you install a local server runtime.
`tools`	Command line tools required for interacting with the cloud runtime (for example, to deploy and start applications) and the local server runtime (for example, to install and start the local server).
`licenses.txt`	Licenses of third-party components contained in the SAP BTP SDK for Neo environment.
`readme.txt`	Brief introduction to the SAP BTP SDK for Neo environment, its content, and how to set it up.
`sdk.version`	SDK version information for use by other tools interacting with the SAP BTP SDK for Neo environment.

Local Server Runtime

The cloud server runtime consists of the application server, the platform API, and the cloud implementations of the provided services (connectivity, SAP HANA and SAP ASE, document, and identity). The SDK, on the other

hand, contains a local server runtime that consists of the same application server, the same platform API, but local implementations of the provided services. These are designed to emulate the cloud server runtime as closely as possible to support the local development and test process.

Supported APIs

The SAP BTP SDK for Neo environment contains the API for SAP BTP. All web applications that should be deployed in the cloud should be compiled against this platform API. The platform API is used by the SAP BTP Tools for Java to set the compile-time classpath.

All JARs contained in the platform API are considered part of the provided scope and must therefore be used for compilation. This means that they must not be packaged with the application, since they are provided and wired at runtime in the SAP BTP runtime, irrespective of whether you run your application locally for development and test purposes or centrally in the cloud.

When you develop applications to run on the SAP BTP, you should be aware of which APIs are supported and provisioned by the runtime environment of the platform:

- Third-party APIs: These include Java EE standard APIs (standards based and backwards compatible as defined in the Java EE Specification) and other APIs released by third parties.
- SAP APIs: The platform APIs provided by the SAP BTP services.

Related Information

Using Samples [page 850]
Console Client for the Neo Environment [page 1348]
API Documentation [page 1147]
Supported Java APIs [page 868]
Deploy Locally with the Console Client [page 890]

1.20.5 SAP BTP SDK for iOS

SAP BTP SDK for iOS enables developers to quickly develop enterprise-ready native iOS apps, built with Swift, the modern programming language by Apple.

The SDK is tightly integrated with SAP BTP mobile service for development and operations to provide:

- End-to-end integrated security
- Support for offline apps
- Enterprise-grade logging and monitoring support
- Access to core SAP S/4HANA data and business processes as well as access to third-party data sources
- Access to SAP BTP capabilities and services

The SDK provides a set of UI controls that are often used in the enterprise space. These controls are implemented using the SAP Fiori design language, and are in addition to the existing native controls on the iOS platform.

Related Information

About SAP BTP SDK for iOS

1.20.6 SAP JVM Tools in Eclipse for the Neo Environment

Features of the SAP JVM Tools in Eclipse for the Neo Environment

You can download SAP JVM Tools in Eclipse from the SAP Development Tools for Eclipse page. The toolkit package contains SAP JVM Profiler and SAP JVM Debugger.

Related Information

Install the SAP JVM Tools in Eclipse [page 846]

1.20.7 Console Client for the Neo Environment

The console client for the Neo environment enables development, deployment and configuration of an application outside the Eclipse IDE as well as continuous integration and automation tasks. The tool is part of the SDK for SAP BTP, Neo environment. You can find it in the *tools* folder of your SDK location.

> **i Note**
>
> The console client is related only to the Neo environment. For the Cloud Foundry environment use the Cloud Foundry command line interface. See Download and Install the Cloud Foundry Command Line Interface.

To learn more about	See
Downloading and setting up the console client	Set Up the Console Client
Opening the tool and working with the commands and parameters	Using the Console Client Console Client Video Tutorial
Groups of console client commands	Console Client Commands

To learn more about	See
Ranges and types of exit codes	Exit Codes
Machine-readable output of commands	Machine-Readable Command Output
Verbose mode of output	Verbose Mode of the Console Commands Output

1.20.8 Working with Cloud Management Tools Feature Set B in the Neo Environment

Enterprise accounts in SAP BTP that have access to cloud management tools feature set B, can also use the enhanced capabilities offered by feature set B with their subaccounts in the Neo environment.

What is cloud management tools feature set B and which enhancements does it offer?

Cloud management tools represent the group of technologies designed for managing SAP BTP.

We're currently renovating and adding core functionalities to SAP BTP. As part of this process, we're migrating accounts from the existing cloud management tools feature set A to the renovated feature set B. We're doing this migration as a phased rollout, so cloud management tools feature sets A and B will coexist for some time.

For general information about the enhancements that are offered with feature set B and how they compare to feature set A, see Cloud Management Tools — Feature Set Overview.

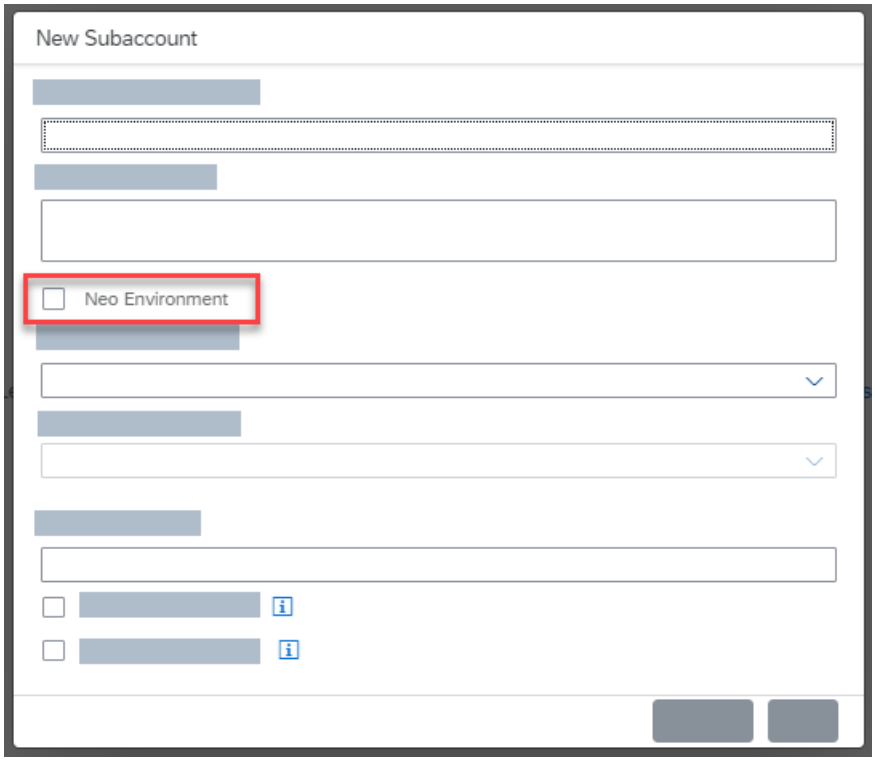
In the rest of this topic, we provide information about the scope of specific enhancements in feature set B when working in the Neo environment.

Is cloud management tools feature set B available to all Neo subaccounts?

Access to cloud management tools feature set B with Neo subaccounts is currently available to enterprise customers on a limited basis within the regions that support the Neo environment.

For information about how to find out which feature set your account has, see Cloud Management Tools — Feature Set Overview.

You can also see if your account has access to the Neo environment by creating a subaccount and checking if the *Neo Environment* option is available in the *New Subaccount* dialog box.



Working with Neo subaccounts in the SAP BTP cockpit

Whenever you directly access a Neo subaccount that supports cloud management tools feature set B, you'll be redirected to a cockpit version that is dedicated to the management of Neo subaccounts. In this cockpit version, you'll notice that:

- The entries in the navigation panel apply only to tools and features that are supported by the Neo environment.
- The *Entitlements* page is for view-only purposes only. To manage entitlements, you must navigate back to the global account.
- When you click on the global account in the navigation breadcrumb or on *SAP BTP, Neo Environment Cockpit* in the cockpit banner, you are redirected to the cockpit version with your global account.

Working with Neo subaccounts using the SAP BTP command line interface

The SAP BTP command line interface (btp CLI) allows you to perform a wide range of account management tasks using the command line. For more information, see Account Administration Using the SAP BTP Command Line Interface (btp CLI) [Feature Set B] [page 1317].

Whenever you directly access a Neo subaccount, you need to work with the Neo command console or the SAP BTP cockpit. These are the known conditions and scope when working with Neo subaccounts in the btp CLI:

btp CLI Command	Conditions and Scope for the Neo Environment
`accounts/environment-instance`	Not applicable. A Neo environment is created with every Neo subaccount. You cannot enable additional environments within a Neo subaccount.
`accounts/resource-provider`	Not applicable. To consume supported remote-service resources that you already own from a non-SAP cloud vendor, use a multi-environment subaccount.

`subscribe accounts/subaccount`
`unsubscribe accounts/subaccount`
`list accounts/subscription`
`get accounts/subscription`
	To manage application subscriptions in Neo subaccounts, use the console client in the SAP BTP SDK for the Neo environment.

`list accounts/available-environment`	Not available for Neo subaccounts.
`list accounts/entitlement`	Not available for Neo subaccounts.
`delete accounts/subaccount`	To delete Neo subaccounts, use the console client in the SAP BTP SDK for the Neo environment.

`security/app`
`security/role`
`security/role-collection`
	Use the APIs in the SAP BTP SDK for the Neo environment.

`services/binding`
`services/broker`
`services/instance`
`services/offering`
`services/plan`
`services/platform`
	These commands apply only to subaccounts that support the multi-environment configuration, such as Cloud Foundry and Kubernetes.

Where necessary, you can use the SAP BTP cockpit to fully manage your Neo environment.

Working with Neo subaccounts and the APIs of the core services for SAP BTP

Feature set B offers a set of core service APIs that allow you to perform a wider range of account management tasks. For more information, see Account Administration Using APIs.

These are the known conditions and scope when working with Neo subaccounts and the core APIs for SAP BTP:

Service	Conditions and Scope for the Neo Environment
Accounts service (provided with the SAP Cloud Management service for SAP BTP)	When deleting Neo subaccounts, use the SAP BTP SDK for the Neo environment.
Provisioning service (provided with the SAP Cloud Management service for SAP BTP)	Use the SAP BTP SDK for the Neo environment.
SAP Software-as-a-Service Provisioning service	To manage application subscriptions in Neo subaccounts, use the SAP BTP SDK for the Neo environment.
SAP Authorization and Trust Management service	Use the SAP BTP SDK for the Neo environment.
SAP Service Manager	The APIs offered by this service apply only to subaccounts that support the multi-environment configuration, such as Cloud Foundry and Kubernetes.
Events	You can use the central-based events provided by this service with your Neo subaccounts. See Using the Events Service APIs.

Where necessary, you can also use the SAP BTP cockpit to fully manage your Neo environment.

Related Information

Cloud Management Tools — Feature Set Overview
Enterprise Accounts
What Is SAP BTP, Neo Environment [page 9]
https://api.hana.ondemand.com/
Console Client Commands [page 1355]
Managing Members Using the Cockpit [page 1294]
https://api.sap.com/api/AccountMembersManagementAPI/resource
Platform Authorization Management API [page 1688]

2 Get an Enterprise Account

To use an enterprise account, you can either purchase a customer account or join the partner program to purchase a partner account.

2.1 Purchase a Customer Account

A customer account is an enterprise account that allows you to host productive, business-critical applications with 24x7 support.

When you want to purchase a customer account, you can select from a set of predefined packages. For information about service availability, prices, and estimators, see https://www.sap.com/products/extension-suite/pricing.html and https://www.sap.com/products/integration-suite/pricing.html. You can also view the service catalog via the SAP Discovery Center. Contact us on SAP BTP or via an SAP sales representative.

In addition, you can upgrade and refine your resources later on. You can also contact your SAP sales representative and opt for a configuration, tailored to your needs.

After you have purchased your customer account, you will receive an e-mail with a link to the Home page of SAP BTP.

Related Information

Commercial Models

2.2 Join the Partner Program

A partner account is an enterprise account that enables you to build applications and to sell them to your customers.

To become a partner, you need to fill in an application form and then sign your partner contract. You will be assigned to a partner account with the respective resources. To apply for the partner program, visit https://partneredge.sap.com/content/partnerregistration/en_us/registration.html?partnertype=BLD&engagement=0002&build=1. You will receive a welcome mail with further information afterwards.

General information about the partner program is available on https://www.sap.com/partner/become/partneredge-build/plan-design.html.

Related Information

Account Model [page 10]

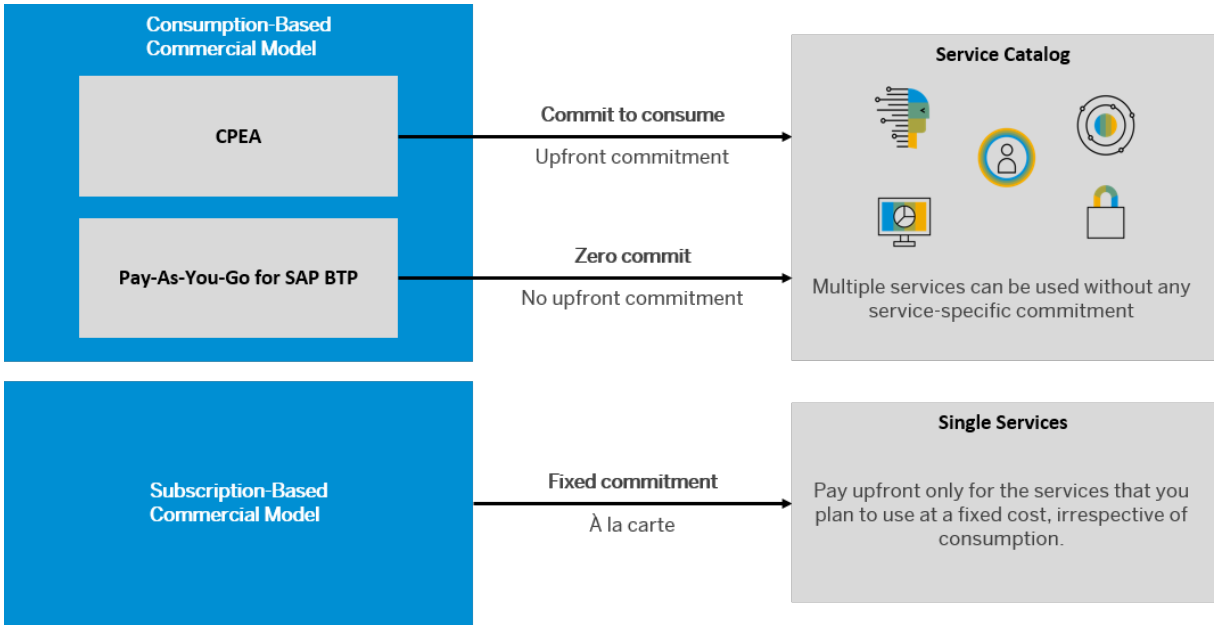
2.3 Commercial Models

SAP BTP offers two different commercial models for enterprise accounts.

- **Consumption-based commercial model:** Your organization receives access to all current and future services that are eligible for this model. You have complete flexibility to turn services on and off and to switch between services as your business requires throughout the duration of your contract. This commercial model is available in two flavors: Cloud Platform Enterprise Agreement (CPEA) and Pay-As-You-Go for SAP BTP.
  For more information, see What Is the Consumption-Based Commercial Model?
- **Subscription-based commercial model:** Your organization subscribes only to the services that you plan to use. You can then use these services at a fixed cost, irrespective of consumption.
  For more information, see What Is the Subscription-Based Commercial Model? [page 827].

For information about service availability, prices, and estimators, see https://www.sap.com/products/extension-suite/pricing.html and https://www.sap.com/products/integration-suite/pricing.html. You can also view the service catalog via the SAP Discovery Center.

> **i Note**
>
> You can use both commercial models, either in separate global accounts or in the same global acccount depending on your business needs. Contact your SAP account executive or sales representative for more information.



2.3.1 What Is the Consumption-Based Commercial Model?

> **i** Note
>
> The use of the consumption-based commercial model is subject to its availability in your country or region.

With the consumption-based model, your organization purchases an entitlement to all current and future SAP BTP services that are eligible for this model. Throughout the duration of your contract, you have complete flexibility to turn services on and off and to switch between services as your business requires.

The consumption-based commercial model is available in two flavors; the **CPEA** (**Cloud Platform Enterprise Agreement**) and **Pay-As-You-Go for SAP BTP**. Each option is suited to different business situations and levels of financial commitment, as described in the table below. For additional information and clarifications, please contact your account executive.

CPEA	• Your organization makes a prepaid investment in cloud credits for the contract duration with an annual commitment to consume SAP BTP services. • This model is suitable for customers that have well-established and planned use cases, and who want the flexibility of turning services on and off, and switching between services, without the commitment of being tied to a single service throughout the duration of the contract. • You receive a monthly balance statement that provides information about the usage consumption of each service and the corresponding costs. The total monthly cost is deducted from your cloud credits balance. • This model has a minimum investment entry, and volume-based discounts are available. • You are billed annually in advance. Any overages are billed in arrears at list price. • You can top up your cloud credits at any time to prevent overages.

Pay-As-You-Go for SAP BTP	• You have the same access to all the services that are available in CPEA, but with a highly flexible zero-commitment model – you pay nothing upfront and there is no minimum usage requirement or annual commitment. • You pay only for the SAP BTP services that you want, when you use them. • You are billed monthly in arrears. • Service charges are non-discountable. • This low-risk model is suitable for customers with use cases that are not well defined, and are interested in running a proof-of-concept in a productive environment. This model provides the flexibility of turning services on and off, and switching between services, as needed throughout the duration of the contract. • A seamless transition to the CPEA model is available, on the condition that you have no other CPEA-based global accounts.

> → Tip
>
> You can monitor costs and service usage throughout the contract period. See Monitoring Usage and Consumption Costs in Your Global Account and View Subaccount Usage Analytics [page 1288].

For information about eligible services and pricing for SAP Integration Suite and SAP Extension Suite, see https://www.sap.com/products/extension-suite/pricing.html and https://www.sap.com/products/integration-suite/pricing.html.see https://cloudplatform.sap.com/price-lists, or access the SAP BTP service catalog via the SAP Discovery Center. The SAP BTP service catalog allows you to identify service availability per data center and to determine licensing model compatibility per available service plan.

As part of the free tier model for SAP BTP, some services offer free service plans that allow you to try out these services without any additional charge within your CPEA or Pay-As-You-Go for SAP BTP global account. For more information, see Using Free Service Plans.

The consumption-based commercial model is available in two flavors; the **CPEA** (**Cloud Platform Enterprise Agreement**) and **Pay-As-You-Go for SAP BTP**. Each option is suited to different business situations and levels of financial commitment, as described in the table below. For additional information and clarifications, please contact your account executive.

To find frequently asked questions about this licensing model, see the Consumption-based commercial model CPEA and Cloud Credits - FAQ Document.

> i Note
>
> Depending on your business needs, you can combine the consumption-based commercial model with the subscription-based commercial model in the same global account, or you can use both commercial models in the same global acccount. Contact your SAP account executive or sales representative for more information.

Related Information

What Is the Subscription-Based Commercial Model? [page 827]
Monitoring Usage and Consumption Costs in Your Global Account
View Subaccount Usage Analytics [page 1288]
Managing Global Accounts Using the Cockpit
Managing Directories Using the Cockpit [Feature Set B] [page 1277]
Managing Subaccounts Using the Cockpit [page 1282]
Managing Entitlements and Quotas Using the Cockpit

2.3.2 What Is the Subscription-Based Commercial Model?

Your organization receives a fixed price and period (typically a 1 to 3-year period) for access to your subscribed SAP BTP services.

Under this commercial agreement:

- You are entitled to use only the subscribed services.
- To access additional services, at an extra cost, you can modify your contract via your sales representative or account executive.
- You pay at a fixed cost, regardless of consumption of subscribed services.
- You pay in advance when the contract period starts.
- Your organization can renew the subscription at the end of the contract period.

For information about available services and pricing, see SAP Store.

You can also access the SAP BTP service catalog via the SAP Discovery Center to identify the availability of services by data center, and also to determine licensing model compatibility per service plan.

Note that some services can be subscribed based on a user metric or a resource metric. For example, a Portal service can be based on the number of site visits or user metrics. A resource-based metric is more common when dealing with a large number of users; for example, suppliers accessing a portal to interact with your organization. Since it isn't always possible to predict how many resources would be required upfront for a three-year period, you can increase your original order if resource usage exceeds your subscribed quota. Using SAP BTP cockpit, you can view resource consumption within your global account on a monthly basis.

In the subscription-based model, you also get access to bundles or packages that comprise several related services and apps. Most of the time, this works out to be more cost effective when compared to subscribing to individual SAP BTP services.

Can an existing global account under a consumption-based contract be transformed to the subscription-based model?

You can change your existing contract from the consumption-based commercial model to a subscription license. Keep in mind that not all services that are eligible for the consumption-based model are compatible

with the subscription-based model. We recommend that you contact your SAP account executive or sales representative to discuss feasibility and terms of transforming your contract.

Note that you can use both consumption and subscription-based commercial models either in separate global accounts or in the same global acccount depending on your business needs. Contact your SAP account executive or sales representative for more information.

Related Information

What Is the Consumption-Based Commercial Model?
Monitoring Usage and Consumption Costs in Your Global Account
View Subaccount Usage Analytics [page 1288]
Managing Global Accounts Using the Cockpit
Managing Directories Using the Cockpit [Feature Set B] [page 1277]
Managing Subaccounts Using the Cockpit [page 1282]
Managing Entitlements and Quotas Using the Cockpit

3 Getting Started, Neo Environment

Get onboarded in the Neo environment of SAP BTP. Follow the workflows for customer accounts or subscribe to business applications.

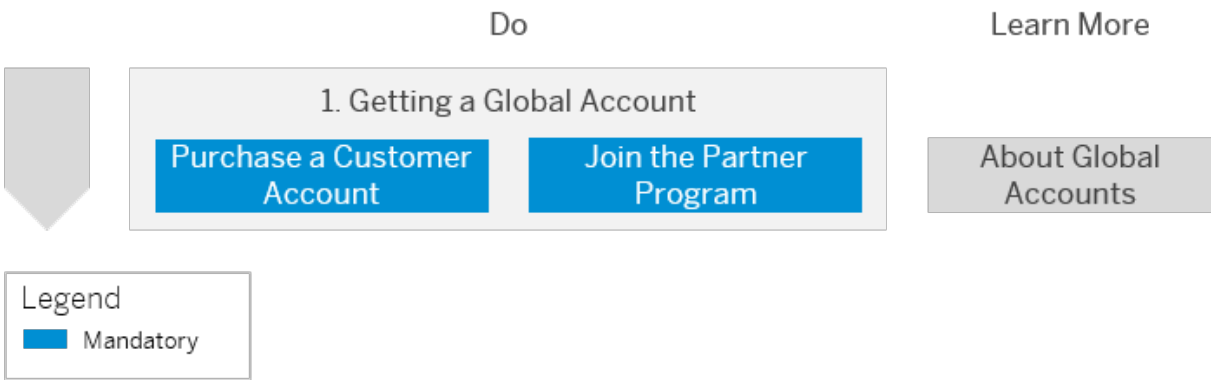
Getting Started with a Customer Account in the Neo Environment [page 829]
: Quickly get started with a customer account.

Getting Started with Business Applications Subscriptions in the Neo Environment [page 833]
: By using SAP BTP, a provider can build and run an application for consumption by multiple consumers. A provider is an SAP partner, who wants to sell business applications to their customers, or an SAP customer, who wants to make their business applications available to different organizational units, for example.

3.1 Getting Started with a Customer Account in the Neo Environment

Quickly get started with a customer account.

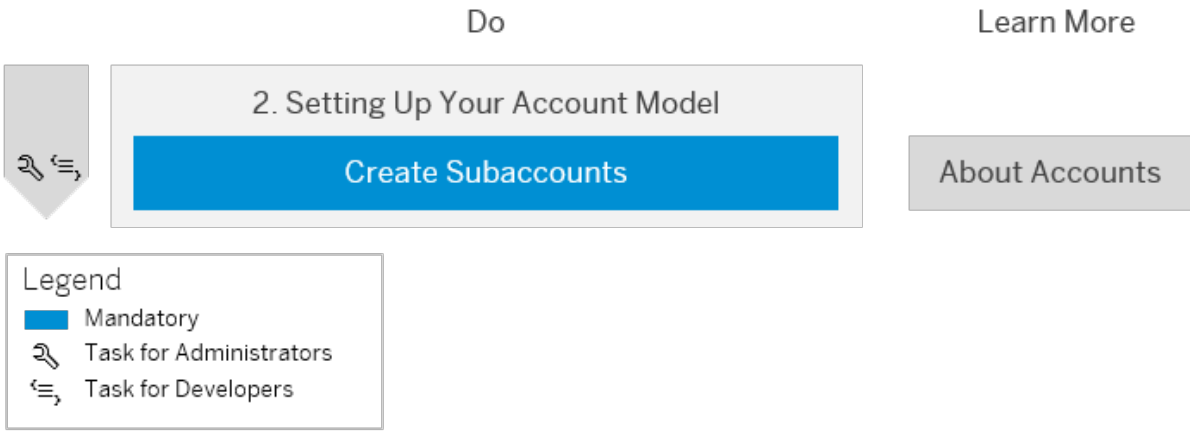
1. Getting a Global Account



- Purchase a Customer Account [page 823]
- Join the Partner Program [page 823]
- Account Model [page 10]

Before you begin, purchase a customer account or join the partner program. See Purchase a Customer Account [page 823] or Join the Partner Program [page 823].

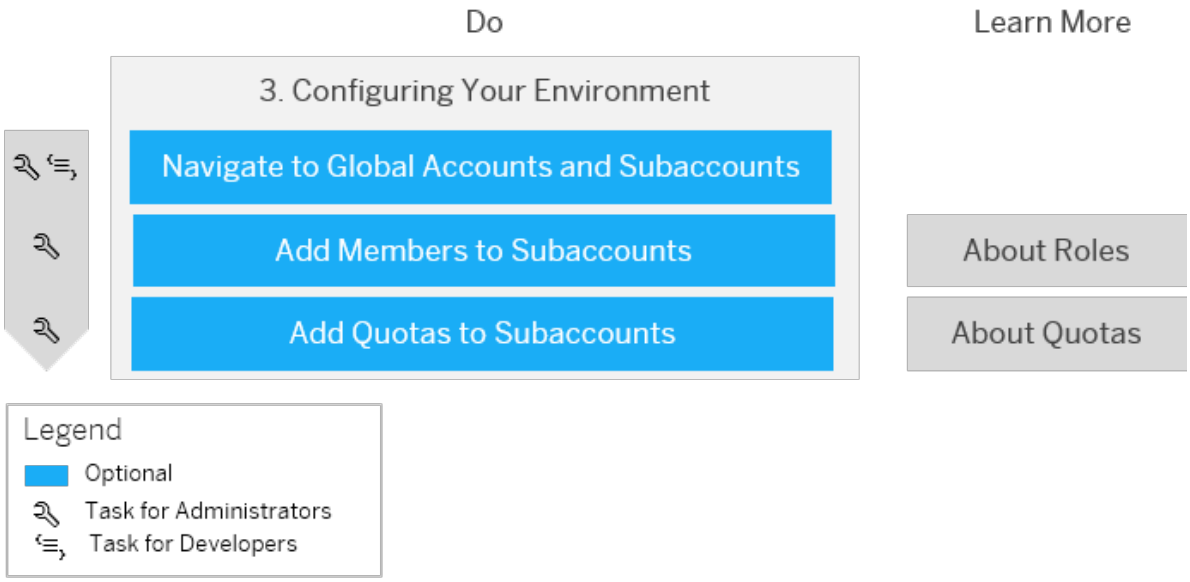
2. Setting Up Your Account Model



- Create a Subaccount
- Global Accounts [page 10]

After you've received your logon data by email, create subaccounts in your global account. This allows you to further break down your account model and structure it according to your business needs. See Create a Subaccount.

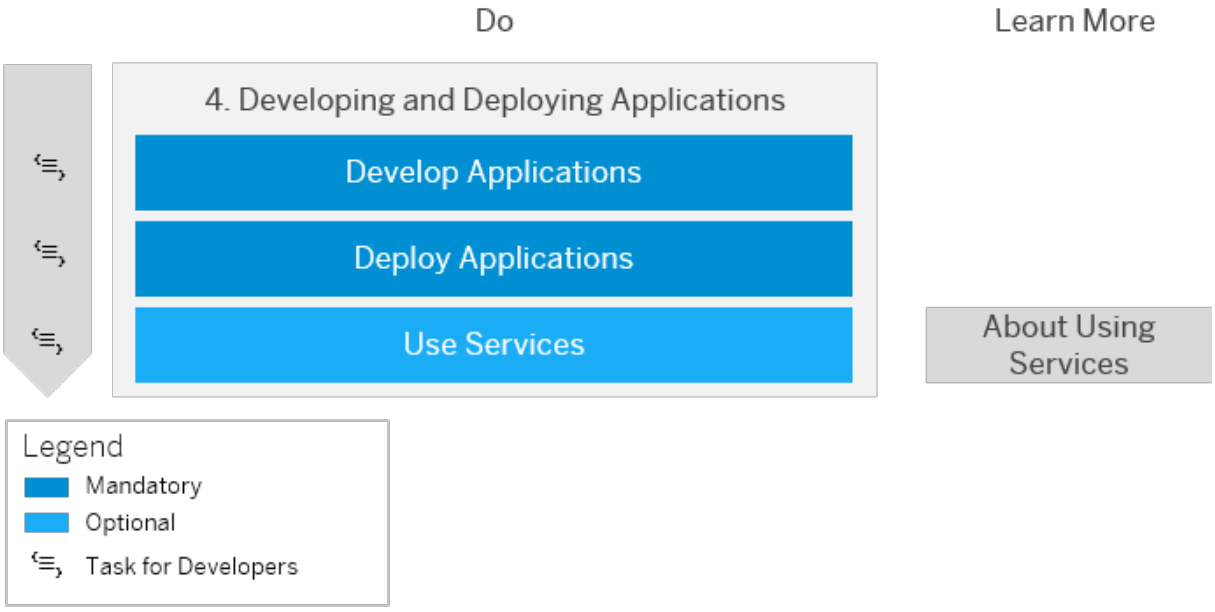
3. Configuring Your Environment



- Navigate in the Cockpit
- Add Members to Your Neo Subaccount [page 1299]
- Configure Entitlements and Quotas for Subaccounts [page 1292]
- Managing Member Authorizations in the Neo Environment [page 1301]
- Configure Entitlements and Quotas for Subaccounts [page 1292]

1. Since you need to use the cockpit to configure your environment, it's important you understand how you can navigate to your global account and subaccounts. See Navigate in the Cockpit.
2. It's time to think about member management. You can add members to subaccounts and assign different roles to those members. For more information, see Add Members to Your Neo Subaccount [page 1299]. For more information about roles, see Managing Member Authorizations in the Neo Environment [page 1301].
3. Before you can start using resources such as application runtimes, you need to manage your entitlements and add quotas to your subaccounts. See Configure Entitlements and Quotas for Subaccounts [page 1292]. To learn more about entitlements and quotas, see Managing Entitlements and Quotas Using the Cockpit [page 1291].

4. Developing and Deploying Applications



- Developing Java in the Neo Environment [page 843]
- Deploying and Updating Applications [page 875]
- Enable Services in the Neo Environment [page 1153]
- Using Services in the Neo Environment [page 1152]

1. Develop and deploy your application. Check out the Developer Guide for tutorials and more information. See Developing Java in the Neo Environment [page 843].
2. Enable a service so that you can integrate it with an application. See Enable Services in the Neo Environment [page 1153].

Parent topic: Getting Started, Neo Environment [page 829]

Related Information

Getting Started with Business Applications Subscriptions in the Neo Environment [page 833]

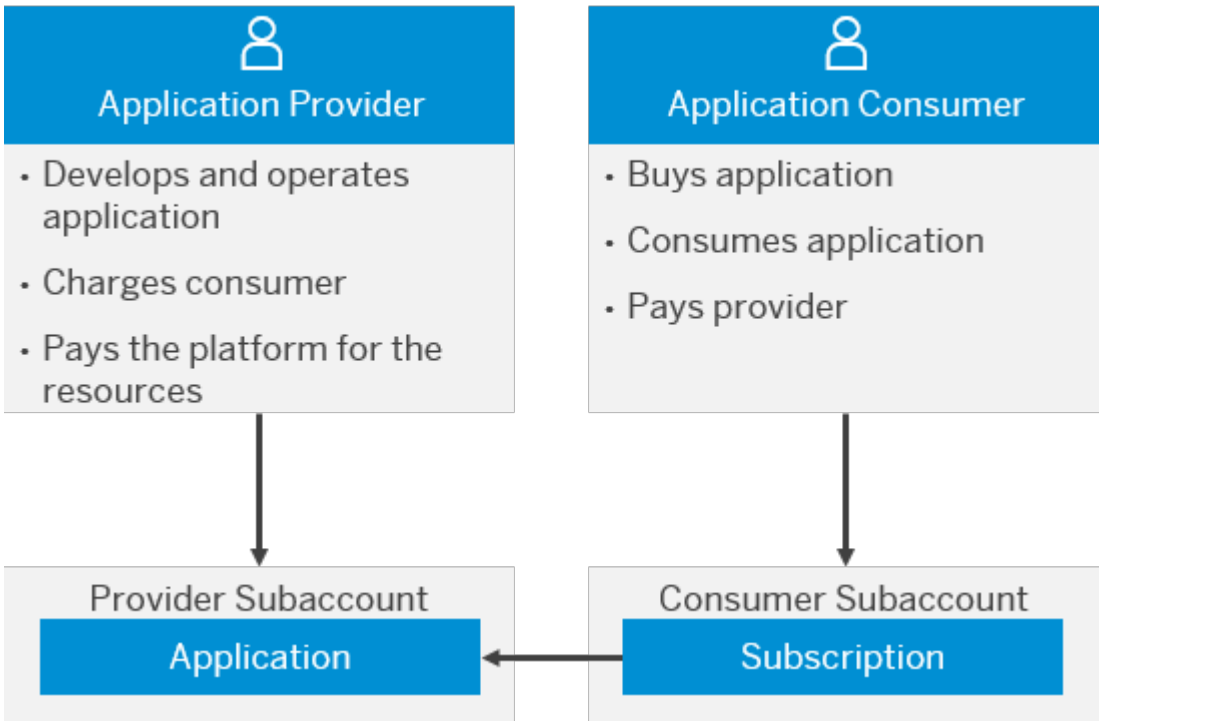
3.2 Getting Started with Business Applications Subscriptions in the Neo Environment

By using SAP BTP, a provider can build and run an application for consumption by multiple consumers. A provider is an SAP partner, who wants to sell business applications to their customers, or an SAP customer, who wants to make their business applications available to different organizational units, for example.

Overview

The platform provides a multitenant functionality, which allows providers to own, deploy, and operate an application for multiple consumers with reduced costs. For example, the provider can upgrade the application for all consumers instead of performing each update individually, or can share resources across many consumers. Application consumers can configure certain application features and launch them using consumer-specific URLs. Furthermore, they can protect the application by isolating their tenants.

Consumers do not deploy applications in their subaccounts, but simply subscribe to the provider application. As a result, a subscription is created in the consumer subaccount. This subscription represents the contract or relation between a subaccount (tenant) and a provider application.



> **i Note**
>
> SAP Partners that wish to offer SAP BTP multitenant business applications in the Cloud Foundry environment should contact SAP.

In the Neo environment, SAP BTP supports Java and HTML5 subscriptions. You configure HTML5 subscriptions used for HTML5 provider applications through the cockpit only. While for Java applications, you

can only use the console client. When such a subscription is set in the consumer subaccount, the Java provider application can use a connectivity destination that you configure in the consumer subaccount.

Multitenancy Roles

The multitenancy concept concerns two main user roles:

- `Application Provider` - an organizational unit that uses SAP BTP to build, run, and sell applications to customers, that is, the application consumers.
  For more information about providing applications, see Providing Multitenant Applications to Consumers in the Neo Environment [page 835].
- `Application Consumer` - an organizational unit, typically a customer or a department inside an organization of a customer, which uses an SAP BTP application for a certain purpose. Obviously, the application is in fact used by end users, who might be employees of the organization (for instance, in the case of an HR application) or just arbitrary users, internal or external (for instance, in the case of a collaborative supplier application).
  For more information about consuming applications, see Subscribe to Java Multitenant Applications in the Neo Environment [page 838] or Subscribe to HTML5 Multitenant Applications in the Neo Environment [page 840].

Subaccounts and Subaccount Members

To use SAP BTP, both the application provider and the application consumer must have a subaccount. The subaccount is the central organizational unit in SAP BTP. It is the central entry point to SAP BTP for both application providers and consumers. It may consist of a set of applications, a set of subaccount members and a subaccount-specific configuration.

Subaccount members are users who are registered via the SAP ID service. Subaccount members may have different privileges regarding the operations that are possible for a subaccount (for example, subaccount administration, deploy, start, and stop applications). Note that the subaccount belongs to an organization and not to an individual. Nevertheless, the interaction with the subaccount is performed by individuals, the members of the subaccount. The subaccount-specific configuration allows application providers and application consumers to adapt their subaccount to their specific environment and needs.

An application resides in exactly one subaccount, the hosting subaccount. It is uniquely identified by the subaccount name and the application name. Applications consume SAP BTP resources, for instance, compute units, structured and unstructured storage and outgoing bandwidth. Costs for consumed resources are billed to the owner of the hosting subaccount, who can be an application provider, an application consumer, or both.

Parent topic: Getting Started, Neo Environment [page 829]

Related Information

Getting Started with a Customer Account in the Neo Environment [page 829]

Providing Multitenant Applications to Consumers in the Neo Environment [page 835]
Providing Java Multitenant Applications to Tenants for Testing [page 836]
Subscribe to Java Multitenant Applications in the Neo Environment [page 838]
Subscribe to HTML5 Multitenant Applications in the Neo Environment [page 840]

3.2.1 Providing Multitenant Applications to Consumers in the Neo Environment

In the Neo environment, you can develop and run multitenant (tenant-aware) applications that you can make available to multiple consumers.

Developing and Deploying Multitenant Applications

For detailed instructions on developing multitenant applications, see Developing Multitenant Applications in the Neo Environment [page 898].

List Java Subscriptions Using the Console Client

Prerequisites

- An enterprise account. For more information, see Global Accounts [page 10].
- Develop and deploy an application in the Neo environment for multiple consumers. For more information, see Developing Multitenant Applications in the Neo Environment [page 898].
- Provider and consumer subaccounts that belong to the same region. For more information, see Regions and Hosts Available for the Neo Environment [page 17].
- Set up the console client. For more information, see Set Up the Console Client [page 847].

To list all subaccounts subscribed to a Java application, use the `list-subscribed-accounts` command.

Example

```
neo list-subscribed-accounts --account mysubaccount --application demo --user
myuser --host us1.hana.ondemand.com
```

3.2.1.1 Providing Java Multitenant Applications to Tenants for Testing

Using the console client, you can create subaccounts and subscribe them to a provider application to test how applications can be provided to multiple consumers.

Prerequisites

- Set up the console client. For more information, see Set Up the Console Client [page 847].
- Develop and deploy an application that is used by multiple consumers. For more information, see Developing Multitenant Applications in the Neo Environment [page 898].
- You have an enterprise account. For more information, see Global Accounts [page 10].
- You are a member to both subaccounts: the one where the multitenant application is deployed and the one that you subscribe to the application.

Subscribe Subaccounts to an Application

Using the console client, you can subscribe subaccounts to an application.

Context

> **i Note**
>
> You can subscribe a subaccount to an application that is running in another subaccount only if both subaccounts (provider and consumer subaccounts) belong to the same region.

Procedure

1. Open the command prompt and navigate to the folder containing `neo.bat/sh (<SDK installation folder>/tools)`.
2. Create subaccounts for several consumers.

   To create a subaccount, execute **`neo create-account -a <subaccount> -u <user name or email> -h <host> -n <display name of the new subaccount>`**.
3. Subscribe the subaccounts you have created to the provider application.

   For each created subaccount, execute **`neo subscribe -a <subaccount> -b <subaccount name>:<application> -u <user name or email> -h <host>`**.

> **Note**
>
> Specify the parameter `-b` in the format <subaccount name>:<application>.

Test the Multitenant Application

Access the application through the different tenants and verify that the multitenant application works as configured for the respective subaccount (tenant).

Procedure

1. Access the application using the dedicated URL for each consumer subaccount in the format **`https://<application name><provider subaccount>-<consumer subaccount>.<host>`**.

   You see the list of subscriptions and the corresponding application URLs to access them in the *Subscriptions* pane in the cockpit.
2. Change the configuration of the multitenant application for each consumer subaccount (tenant).
3. Verify that the configuration of the provider application differs for each consumer subaccount (tenant).
4. (Optional) You can also check the list of your test subaccounts and subscriptions as follows:

Option	Description
List all applications to which a given subaccount is subscribed	Execute **`neo list-subscribed-applications -a <subaccount> -u <user name or email> -h <host>`**.
List subaccounts you have created	Execute **`neo list-accounts -a <subaccount> -u <user name or email> -h <host>`**.
List all subaccounts subscribed to a given application	Execute **`neo list-subscribed-accounts -a <subaccount> -b <application> -u <user name or email> -h <host>`**.

Clean Up Your Environment

Procedure

1. Unsubscribe each consumer subaccount from the application.

   For each consumer subaccount, execute **`neo unsubscribe -a <subaccount> -b <application> -u <user name or email>`**.
2. Delete each consumer subaccount.

   For each consumer subaccount, execute **`neo delete-account -a <subaccount> -h <host> -u <user name or email>`**.

Related Information

create-account [page 1379]
list-accounts [page 1478]
subscribe [page 1566]
list-subscribed-accounts [page 1507]
list-subscribed-applications [page 1509]
delete-account [page 1400]
unsubscribe [page 1577]

3.2.2 Subscribe to Java Multitenant Applications in the Neo Environment

Create, list, and remove subscriptions for a Java application using the console client and view all our subscriptions in the cockpit.

Prerequisites

- An enterprise account. For more information, see Global Accounts [page 10].
- Develop and deploy an application in the Neo environment for multiple consumers. For more information, see Developing Multitenant Applications in the Neo Environment [page 898].
- Provider and consumer subaccounts that belong to the same region. For more information, see Regions and Hosts Available for the Neo Environment [page 17].
- If applicable, purchase SaaS licenses for the applications you want to consume.Set up the console client. For more information, see Set Up the Console Client [page 847].

Create Java Subscriptions

To create Java subscriptions, use the `subscribe` command.

Example

```
neo subscribe --account consumersubaccount --application mysubaccount:myapp --
user myuser --host us1.hana.ondemand.com
```

List Java Subscription in the Console Client

- To list all Java applications subscribed to a subaccount, use the `list-subscribed-applications` command.

  Example

  ```
  neo list-subscribed-applications --account consumersubaccount --user
  myuser --host hana.ondemand.com
  ```

- To list all subaccounts subscribed to a Java application, use the `list-subscribed-accounts` command.

  Example

  ```
  neo list-subscribed-accounts --account mysubaccount --application demo --
  user myuser --host us1.hana.ondemand.com
  ```

View Java Subscriptions in the Cockpit

Procedure

1. Open the subaccount in the cockpit.
2. In the navigation area, choose *Applications* > *Subscriptions*.

   You see a list of subscriptions to Java applications, with the provider subaccount from which the subscription was obtained and the subscribed application.
3. To navigate to the subscription overview, choose the application name. You have the following options:
   - To launch an application, choose the link in the *Application URLs* panel.
   - To create connectivity destinations, choose *Destinations* in the navigation area.
   - To create or assign roles, choose *Roles* in the navigation area.

Remove Java Subscriptions

> **Note**
>
> Some subscriptions automatically created by the platform cannot be removed.

To remove Java subscriptions, use the `unsubscribe` command.

Example

```
neo unsubscribe --account consumersubaccount --application mysubaccount:myapp
--user myuser --host us1.hana.ondemand.com
```

Related Information

Subscribe to HTML5 Multitenant Applications in the Neo Environment [page 840]
list-subscribed-applications [page 1509]
list-subscribed-accounts [page 1507]
subscribe [page 1566]
unsubscribe [page 1577]

3.2.3 Subscribe to HTML5 Multitenant Applications in the Neo Environment

Manage subscriptions to HTML5 applications by viewing, creating, or removing subscriptions in the cockpit.

Create HTML5 Subscriptions

Context

Procedure

1. Open the subaccount in the cockpit.

   For more information, see Navigate in the Cockpit.
2. In the navigation area, choose *Applications* > *Subscriptions*.
3. On the *Subscribed HTML5 Applications* panel in the *Subscriptions* section, choose *New Subscription*.
4. Select the provider subaccount. You can select subaccounts that provided applications to your subaccount as well as subaccounts where the current user has the administrator role.
5. Select the application to which you want to subscribe.
6. Enter a subscription name.

> **i Note**
>
> The subscription name must be unique across all subscription names and all HTML5 application names in the current subaccount.

View HTML5 Subscriptions

Context

Procedure

1. In the navigation area, choose *Applications* > *Subscriptions*. The subscriptions to HTML5 applications are listed with the following information:
   - The subaccount name of the application provider from which the subscription was obtained
   - The name of the subscribed application
2. To navigate to the subscription overview, click the application name:
   - To launch an application, click the URL link in the *Active Version* panel.
   - To create or assign roles, choose *Roles* in the navigation area.

Remove HTML5 Subscriptions

Context

Procedure

1. Choose the *Unsubscribe* icon.
2. Confirm your operation.

Related Information

Subscribe to Java Multitenant Applications in the Neo Environment [page 838]

4 Development, Neo Environment

Learn more about developing applications on the Neo environment of SAP BTP.

Overview

The Neo environment of SAP BTP enables you to develop and run cloud applications using technologies such as Java EE, SAP HANA, HTML5, SAPUI5, and so on. The environment itself is SAP-proprietary but it is designed to run applications based on community or SAP technologies.

Deploy Business Applications in the Neo Environment [page 842]

Developing Java in the Neo Environment [page 843]
SAP BTP enables you to develop, deploy and use Java applications in a cloud environment. Applications run on a runtime container where they can use the platform services APIs and Java EE APIs according to standard patterns.

Developing SAP HANA in the Neo Environment [page 983]
With SAP BTP, you can use the SAP HANA development tools to create comprehensive analytical models and build applications with SAP HANA's programmatic interfaces and integrated development environment.

Multitarget Applications for the Neo Environment [page 1016]
A Multitarget Application (MTA) is a package comprised of multiple application and resource modules, which have been created using different technologies and deployed to different runtimes, but have a common lifecycle. You bundle the modules together, describe them along with their interdependencies to other modules, services, and interfaces, and package them in an MTA.

Developing HTML5 Applications in the Neo Environment [page 1117]
SAP BTP enables you to easily develop and run HTML5 applications in a cloud environment.

API Documentation [page 1147]
API documentation for the Neo environment.

Using Services in the Neo Environment [page 1152]
Learn more about using services in the Neo environment:

4.1 Deploy Business Applications in the Neo Environment

To deploy business applications bundled in a Multi-Target Applications (MTA) archive, use one of the following options:

- The `deploy-mta` command for the Command Line Interface (CLI), as described in deploy-mta [page 1429].

- The *Solutions* view of the SAP BTP cockpit, as described in Deploying Solutions Using the Cockpit [page 1092].

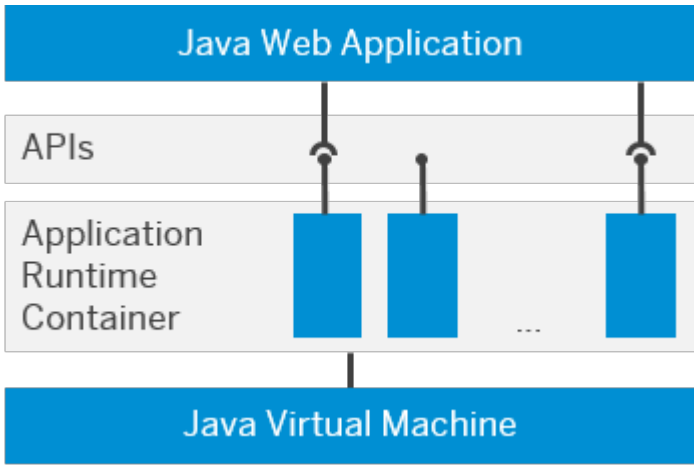
Related Information

deploy-mta [page 1429]
Deploying Solutions Using the Cockpit [page 1092]

4.2 Developing Java in the Neo Environment

SAP BTP enables you to develop, deploy and use Java applications in a cloud environment. Applications run on a runtime container where they can use the platform services APIs and Java EE APIs according to standard patterns.

The SAP BTP Runtime for Java enables the provisioning and running applications on the platform. The runtime is represented by Java Virtual Machine, Application Runtime Container and Compute Units. Cloud applications interact at runtime with the containers and services via the platform APIs.



Compute Unit

The Java development process is enabled by the SAP BTP Tools, which include the SAP BTP SDK.

During and after development, you can configure and operate an application using the cockpit and the console client.

Benefits and advantages

- Offers standardized environment
- Supports a wide-spread Apache Tomcat Web container
- Supports the platform services APIs

Appropriate for

- Developing and running Java Web applications based on standard JSR APIs

- Executing Java Web applications which include third-party Java libraries and frameworks supporting standard JSR APIs
- Supporting Apache Tomcat Java Web applications.

Not appropriate for

- Applications featuring modifications of JSR APIs and their implementations
- Applications requiring modified or customized Java Apache Tomcat Web container

Related Information

Java: Getting Started [page 844]
Runtime for Java [page 854]
Development Environment [page 868]
Developing Java Applications [page 871]
Supported Java APIs [page 868]

4.2.1 Java: Getting Started

Set up your Java development environment and deploy your first application in the cloud.

1. Getting a Global Account
   You first need to get an SAP BTP global account.
   Then you have to create a subaccount in the global account.
2. Set Up
   Download and unzip the SAP BTP SDK for the Neo environment.
3. Create or Import
   Create a classic Java EE application or import an existing sample application to get started.
4. Deploy
   Deploy your application using the console client in the Neo environment.
5. Monitor
   You can view the status and logs of your Java applications.

Samples

A set of sample applications allows you to explore the core functionality of SAP BTP and shows how this functionality can be used to develop complex Web applications. See: Using Samples [page 850]

4.2.1.1 Setting Up the Development Environment

Before you can start developing your application, you need to download and set up the necessary tools, SAP BTP Tools, and SDK.

SAP BTP Tools, SAP BTP SDK for Neo environment, SAP JVM, and the Cloud Connector, can be downloaded from the SAP Development Tools page.

For more information on each step of the set up procedure, open the relevant page from the structure.

Procedure

1. For Java applications, choose between three types of SAP BTP SDK for Neo environment.
   For more information, see Install the SAP BTP SDK for Neo Environment [page 845].
2. SAP JVM is the Java runtime used in SAP BTP. It can be set as a default JRE for your local runtime.
   For instructions on how to install it, see (Optional) Install SAP JVM [page 846].
3. To configure the Console Client, see Set Up the Console Client [page 847].
4. If you need to establish a connection between on-demand applications in SAP BTP and existing on-premise systems, you can use the Cloud Connector.
   For more information, see Cloud Connector.

4.2.1.1.1 Install the SAP BTP SDK for Neo Environment

Context

SAP BTP offers several SDKs for Java development.

For more information, see section Application Runtime Container [page 857].

Procedure

1. Open https://tools.hana.ondemand.com/#cloud
2. From the *SAP BTP SDK for Neo environment* section, download the relevant ZIP file and save it to your local file system.
3. Extract the ZIP file to a folder on your computer or network.

Your SDK is ready for use. To use the SAP BTP SDK for Neo environment with the console client, see Using the Console Client [page 1348].

Related Information

Application Runtime Container [page 857]

4.2.1.1.2 Install the SAP JVM Tools in Eclipse

Context

SAP JVM Tools for Eclipse is an Eclipse plugin that allows you to debug and profile applications which run on SAP JVM. The SAP JVM Profiler and SAP JVM Debugger are included in the SAP JVM Tools.

> **i** Note
>
> Use Eclipse Oxygen or newer version of Eclipse IDE.

Procedure

1. Open Eclipse and go to *Help* > *Install New Software* .
2. In the *Work with* field, enter `https://tools.hana.ondemand.com/oxygen` and choose *SAP BTP Tools*.
3. Choose *Next* and follow the installation wizard.

4.2.1.1.3 (Optional) Install SAP JVM

Context

SAP BTP infrastructure runs on SAP's own implementation of a Java Virtual Machine - SAP Java Virtual Machine (JVM).

SAP JVM is a certified Java Virtual Machine and Java Development Kit (JDK), compliant to Java Standard Edition (SE) 8. Technology-wise it is based on the OpenJDK and has been enhanced with a strong focus on supportability and reliability. One example of these enhancements is the SAP JVM Profiler. The SAP JVM Profiler is a tool that helps you analyze the resource consumption of a Java application running on theSAP BTP local runtime. You can use it to profile simple stand-alone Java programs or complex enterprise applications.

Customer support is provided directly by SAP for the full maintenance period of SAP applications that use the SAP JVM. For more information, see Java Virtual Machine [page 855]

Follow the steps below to install an SAP Java Virtual Machine.

> **Note**
>
> This is an optional procedure. You can also run your local server for SAP BTP on a standard JDK platform, that is an Oracle JVM. SAP JVM, however, is a prerequisite for local profiling with the SAP JVM Profiler.

Procedure

1. Open https://tools.hana.ondemand.com/#cloud
2. From the *SAP JVM* section, download the SAP JVM archive file compatible to your operating system and save it to your local file system.
3. Extract the archive file.

> **Note**
>
> If you use Windows as your operating system, you need to install the Visual C++ 2013 Runtime prior to using SAP JVM. The installation package for the Visual C++ 2013 Runtime can be obtained from Microsoft. Download and install *vcredist_x64.exe* from the following site: https://www.microsoft.com/de-de/download/details.aspx?id=40784 . Even if you already have a different version of Visual C++ Runtime, for example Visual C++ 2015, you still need to install Visual C++ 2013 Runtime prior to using SAP JVM. See SAP Note 2676219.

Related Information

4.2.1.1.4 Set Up the Console Client

Prerequisites

You have downloaded and extracted the SAP BTP SDK for Neo environment. For more information, see Install the SAP BTP SDK for Neo Environment [page 845].

Context

SAP BTP console client is part of the SAP BTP SDK for Neo environment. You can find it in the `tools` folder of your SDK installation. Before using the tool, you need to configure it to work with the platform.

Procedure

1. Open the command prompt.
2. Change the current directory to the `<SDK_installation_folder>\tools` location, which contains the *neo.bat* and *neo.sh* files. For example:

```
cd C:\HCP\SDK
cd tools
```

3. In case you use a proxy server, specify the proxy settings by using environment variables. You can find sample proxy settings in the *readme.txt* file in the `\tools` folder of your SDK location.
   - `Microsoft Windows`

     > **i** Note
     >
     > - For the new variables to be effective every time you open the console, define them using *Advanced System Settings* > *Environment Variables* and restart the console.
     > - For the new variables to be valid only for the currently open console, define them in the console itself.

     For example, if your proxy host is **`proxy`** and proxy port is **`8080`**, specify the following environment variables:

```
set HTTP_PROXY_HOST=proxy
set HTTP_PROXY_PORT=8080
set HTTPS_PROXY_HOST=proxy
set HTTPS_PROXY_PORT=8080
set HTTP_NON_PROXY_HOSTS="localhost"
```

     If you need basic proxy authentication, enter your user name and password:

```
set HTTP_PROXY_USER=<user_name>
set HTTP_PROXY_PASSWORD=<password>
set HTTPS_PROXY_USER=<user_name>
set HTTPS_PROXY_PASSWORD=<password>
```

   - `Linux`, `Mac OS X`, or other `Unix` based OS:

```
export http_proxy=http://proxy:8080
export https_proxy=https://proxy:8080
export no_proxy="localhost"
```

     If you need basic proxy authentication, enter your user name and password:

```
export http_proxy=http://user:password@proxy:8080
export https_proxy=https://user:password@proxy:8080
```

Related Information

Console Client for the Neo Environment [page 1348]

4.2.1.2 Updating SAP BTP SDK for Neo Environment

If you have already installed and used the SAP BTP Tools, SAP BTP SDK for Neo environment and SAP JVM, you only need to keep them up to date.

Follow the relevant procedures listed below.

- sdk-upgrade [page 1538]
-

4.2.1.2.1 Update the SAP BTP SDK for Neo Environment

Context

If you have already installed an SAP BTP SDK for Neo environment package, you only need to update it regularly. To update your SDK, follow the steps below.

Procedure

1. Download the new SAP BTP SDK for Neo environment version from https://tools.hana.ondemand.com/#cloud
2. Unzip the SDK to a new directory on your local file system. Do not install the new SDK version to a directory that already contains SDK.
3. Go to the *Servers* tab view.
4. Stop and delete all local servers.
5. Choose *Window* > *Preferences* > *Server* > *Runtime Environment*.
   For each previously added local runtime:
   1. Select the corresponding entry in the table.
   2. Choose the *Edit* button.
   3. Locate the new SDK version:
      - For **`Java Web`**: Select option *Use Java Web SDK from the following location* and then choose the *Browse* button and find the folder where you have unpacked the SDK ZIP file.
      - For **`Java Web Tomcat 7`**: Choose the *Browse* button and find the folder where you have unpacked the SDK ZIP file or use the *Download and Install* button to get the latest version.
      - For **`Java Web Tomcat 8`**: Choose the *Browse* button and find the folder where you have unpacked the SDK ZIP file or use the *Download and Install* button to get the latest version.
      - For **`Java EE 6 Web Profile`**: Select option *Use Java EE 6 Web Profile SDK from the following location* and then choose the *Browse* button and find the folder where you have unpacked the SDK ZIP file.

> **i Note**
>
> Again, if the SAP BTP SDK for Neo environment version is higher and not supported by the version of your SAP BTP Tools for Java, a message appears prompting you to update your SAP BTP Tools for Java. You can check for updates (recommended) or ignore the message.

4. Choose *Finish*.

6. After editing all local runtimes, choose *OK*.

Related Information

Install the SAP BTP SDK for Neo Environment [page 845]
Application Runtime Container [page 857]
sdk-upgrade [page 1538]

4.2.1.3 Using Samples

The sample applications allow you to explore the core functionality of SAP BTP and show how this functionality can be used to develop more complex Web applications. The samples are included in the SAP BTP SDK for Neo environment or presented as blogs in the SAP Community.

SDK Samples

The samples provided as part of the SAP BTP SDK for Neo environment introduce important concepts and application features of the SAP BTP and show how common development tasks can be automated using build and test tools.

The samples are located in the `<sdk>/samples` folder. The table below lists the samples currently available:

Sample	Feature	More Information
authentication	HTTP BASIC authentication scheme	User Authentication [page 1678]
connectivity	Consumption of Internet services	Consume Internet Services (Java Web or Java EE 6 Web Profile) [page 155]
persistence-with-ejb	Container-managed persistence with JPA	Tutorial: Adding Container-Managed Persistence with JPA (SDK for Java EE 7 Web Profile) [page 918]
persistence-with-jpa	Application-managed persistence with JPA	Tutorial: Adding Application-Managed Persistence with JPA (SDK for Java Web) [page 931]
persistence-with-jdbc	Relational persistence with JDBC	Tutorial: Adding Persistence with JDBC (SDK for Java Web) [page 966]
mail	Sending e-mails	Example: Send E-Mails [page 211]

Sample	Feature	More Information
SAP_Jam_OData_HCP	Accessing data in SAP Jam via OData	Source code for using the SAP Jam API

All samples can be imported as Maven projects. While the focus has been placed on the Apache Maven tools due to their wide adoption, the principles apply equally to other IDEs and build systems.

For more information about using the samples, see Building Samples with Maven [page 852].

Community Samples: Paul the Octopus

The Web application "Paul the Octopus" is part of a community blog and shows how the SAP BTP services and capabilities can be combined to build more complex Web applications, which can be deployed on the SAP BTP.

Features of "Paul the Octopus":

- It is intended for anyone who would like to gain hands-on experience with the SAP BTP.
- It involves the following platform services: identity, connectivity, SAP HANA and SAP ASE, and document.
- Its user interface is developed via SAPUI5 and is based on the Model-View-Controller concept. SAPUI5 is based on HTML5 and can be used for building applications with sophisticated UI. Other technologies that you can see in action in "Paul the Octopus" are REST services and job scheduling.

For more information, see the SAP Community blog: Get Ready for Your Paul Position.

You can get the application source from https://github.com/SAP/cloud-paulpredicts/ .

Community Samples: SAP Library

The Web application "SAP Library" is presented in a community blog as another example of demonstrating the usage of several SAP BTP services in one integrated scenario, closely following the product documentation. You can import it as a Maven project, play around with your own library, and have a look at how it is implemented. It allows you to reserve and return books, edit details of existing ones, add new titles, maintain library users' profiles and so on.

Features of "SAP Library":

- The library users authenticate using the identity service. It supports Single Sign-On (SSO).
- The books' status and features are persisted using the SAP HANA and SAP ASE service.
- Book's details are retrieved using a public Internet Web service, demonstrating usage of the connectivity service.
- The e-mails you will receive when reserving and returning books to the library, are implemented using a Mail destination.
- When you upload your profile image, it is persisted using the document service.

For more information, see the SAP Community blog: Welcome to the Library!

You can get the application source from https://github.com/SAP/cloud-sample-library/

Related Information

Building Samples with Maven [page 852]
Building Java Web Applications with Maven
Working with the "Neo" Maven Plugin

4.2.1.3.1 Building Samples with Maven

All samples provided can be built with Apache Maven. The Maven build shows how a headless build and test can be completely automated.

Context

The build and test does the following:

- Builds a Java Web application based on the SAP BTP API
- Demonstrates how to run rudimentary unit tests (not available in all samples)
- Installs, starts, waits for, and stops the local server runtime
- Deploys the application to the local server runtime and runs the integration test
- Starts, waits for, and stops the cloud server runtime
- Deploys the application to the cloud server runtime and runs the integration test

Related Information

Build Samples from the Command Line [page 853]
Building Java Web Applications with Maven
Working with the "Neo" Maven Plugin

4.2.1.3.1.1 Build Samples from the Command Line

You can use the Apache Maven command line tool to run local and cloud integration tests for any of the SDK samples.

Prerequisites

- You have downloaded the Apache Maven command line tool. For more information, see the detailed Maven documentation at http://maven.apache.org .
- You are familiar with the Maven build lifecycle. For more information, see http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html .

Context

Procedure

1. Open the folder of the relevant project, for example, `<sdk>/samples/hello-world`, and then open the command prompt.
2. Enter the `verify` command with the following profile in order to activate the local integration test:

   ```
   mvn clean verify -P local-integration-tests ...
   ```

   If you are using a proxy, you need to define additional Maven properties as described below in step 4 (see proxy details).
3. Press `ENTER` to start the build process.
   All phases of the default lifecycle are executed up to and including the verify phase, with the resulting build status shown on completion.
4. To activate the cloud integration test, which involves deploying the built Web application on a landscape in the cloud, enter the following profile with the additional Maven properties given below:

   ```
   mvn clean verify -P cloud-integration-tests ...
   ```

   - Landscape host
     The landscape host (default: `hana.ondemand.com`) is predefined in the parent `pom.xml` file (`<sdk>/samples/pom.xml`) and can be overwritten, as necessary. If you have a developer account, for example, and are therefore using the trial landscape, enter the following:

     ```
     mvn clean verify -P cloud-integration-tests -
     Dsap.cloud.host=hanatrial.ondemand.com ...
     ```
   - Account details

Provide your account, user name, and password:

```
mvn clean verify -P cloud-integration-tests -Dsap.cloud.account=<account> -
Dsap.cloud.username=<user name> -Dsap.cloud.password=<password> ...
```

- Proxy details
  If you use a proxy for HTTPS Internet access, provide your proxy host (`https.proxyHost`) and if necessary your proxy port (`https.proxyPort`):

  ```
  mvn clean verify -P cloud-integration-tests -Dhttps.proxyHost=<proxy host>
  -Dhttps.proxyPort=<proxy port> ...
  ```

  > → Tip
  >
  > If your proxy requires authentication, you might want to use the Authenticator class to pass the proxy user name and password. For more information, see Authenticator . Note that for the sake of simplicity this feature has not been included in the samples.

> → Tip
>
> To avoid having to repeatedly enter the Maven properties as described above, you can add them directly to the `pom.xml` file, as shown in the example below:
>
> ```
> <sap.cloud.username>p0123456789</sap.cloud.username>
> ```
>
> You might also want to use environment variables to set the property values dynamically, in particular when handling sensitive information such as passwords, which should not be stored as plain text:
>
> ```
> <sap.cloud.password>${env.SAP_CLOUD_PASSWORD}</sap.cloud.password>
> ```

Related Information

Regions and Hosts Available for the Neo Environment [page 17]

4.2.2 Runtime for Java

The SAP BTP Runtime for Java comprises the components which create the environment for provisioning and running applications on SAP BTP. The runtime is represented by Java Virtual Machine, Application Runtime Container and Compute Units. Cloud applications can interact at runtime with the containers and services via the platform APIs.

Components

- Java Virtual Machine

SAP BTP infrastructure runs on SAP's own implementation of a Java Virtual Machine - SAP Java Virtual Machine (JVM). SAP JVM is a certified Java Virtual Machine and Java Development Kit (JDK), compliant to Java Standard Edition (SE) 8. Technology-wise it is based on the OpenJDK and has been enhanced with a strong focus on supportability and reliability.

- Application Runtime Container
  Applications developed on SAP BTP run on a modular and lightweight runtime container which allows them to consume standard Java EE APIs and platform services that are centrally provided and shared across the platform. SAP BTP leverages open source as a key element and Java EE 6 Web Profile as a default programming model.
- Compute Units
  A Compute Unit is a virtualized hardware on which a SAP BTP application runs.

Related Information

Java Virtual Machine [page 855]
Application Runtime Container [page 857]
Compute Units [page 867]
Supported Java APIs [page 868]

4.2.2.1 Java Virtual Machine

SAP BTP infrastructure runs on SAP's own implementation of a Java Virtual Machine - SAP Java Virtual Machine (JVM).

SAP JVM is a certified Java Virtual Machine and Java Development Kit (JDK), compliant to Java Standard Edition (SE) 8. Technology-wise it is based on the OpenJDK and has been enhanced with a strong focus on supportability and reliability. One example of these enhancements is the SAP JVM Profiler. The SAP JVM Profiler is a tool that helps you analyze the resource consumption of a Java application running on theSAP BTP local runtime. You can use it to profile simple stand-alone Java programs or complex enterprise applications.

SAP JVM

The SAP JVM is a standard compliant certified JDK, supplemented by additional supportability and developer features and extensive monitoring and tracing information. All these features are designed as interactive, on-demand facilities of the JVM with minimal performance impact. They can be switched on and off without having to restart the JVM (or the application server that uses the JVM).

Supportability and Developer Features

Debugging on Demand

With SAP JVM debugging on demand, Java developers can activate and deactivate Java debugging directly – there is no need to start the SAP JVM (or the application server on top of it) in a special mode. Java debugging in the SAP JVM can be activated and deactivated using the *jvmmon tool*, which is part of the SAP JVM delivery. This feature does not lower performance if debugging is turned off. The SAP JVM JDK is delivered with full source code providing debugging information, making Java debugging even more convenient.

Profiling

To address the root cause of all performance and memory problems, the SAP JVM comes with the SAP JVM Profiler, a powerful tool that supports the developer in identifying runtime bottlenecks and reducing the memory footprint. Profiling can be enabled on-demand without VM configuration changes and works reliably even for very large Java applications.

The user interface – the SAP JVM Profiler – can be easily integrated into any Eclipse-based environment by using the established plug-in installation system of the Eclipse platform. It allows you to connect to a running SAP JVM and analyze collected profiling data in a graphical manner. The profiler plug-in provides a new perspective similar to the debug and Java perspective.

A number of profiling traces can be enabled or disabled at any point in time, resulting in snapshots of profiling information for the exact points of interest. The SAP JVM Profiler helps with the analysis of this information and provides views of the collected data with comprehensive filtering and navigation facilities.

The profiling traces provided address the following use cases:

- Memory Allocation Analysis – investigates the memory consumption of your Java application and finds allocation hotspots
- Performance Analysis – investigates the runtime performance of your application and finds expensive Java methods
- Network Trace - analyzes the network traffic
- File I/O Trace - provides information about file operations
- Synchronization Trace - detects synchronization issues within your application
- Method Parameter Trace – yields detailed information about individual method calls including parameter values and invocation counts
- Profiling Lifecycle Information – a lightweight monitoring trace for memory consumption, CPU load, and GC events.

Extensive Monitoring and Tracing Information

The SAP JVM provides comprehensive statistics about threads, memory consumption, garbage collection, and I/O activities. For solving issues with SAP JVM, a number of traces may be enabled on demand. They provide additional information and insight into integral VM parts such as the class loading system, the garbage

collection algorithms, and I/O. The traces in the SAP JVM can be switched on and off using the *jvmmon tool*, which is part of the SAP JVM delivery.

Further Information

Critical Java exceptions (such as NullPointerException, ClassCastException, NoClassDefFoundError, or OutOfMemoryError) provide additional information and further details to help identify the reason for an exception.

Thread dumps not only contain a Java execution stack trace, but also information about monitors or locks, consumed CPU and memory resources, I/O activities, and a description of communication partners (in the case of network communication).

For more information about Java SE Technologies in SAP Products, see 2700275.

Related Information

(Optional) Install SAP JVM [page 846]

4.2.2.2 Application Runtime Container

SAP BTP applications run on a modular and lightweight application runtime container where they can use the platform services APIs and Java EE APIs according to standard patterns.

Depending on the runtime type and corresponding SDK you are using, SAP BTP provides the following profiles of the application runtime container:

Profile	Provides support for	Supported Java versions	Use
Java EE 7 Web Profile TomEE 7 [page 859]	Java EE 7 Web Profile APIs	8	If you need an application runtime container together with all containers defined by the Java EE 7 Web Profile specification.
Java Web Tomcat 9 [page 861]	Some of the standard Java EE 7 APIs (Servlet, JSP, EL, Websocket)	8 (default); 11	If you need a simplified Java Web application runtime container based on Apache Tomcat 9.
Java Web Tomcat 8 [page 862]	Some of the standard Java EE 7 APIs (Servlet, JSP, EL, Websocket)	8	If you need a simplified Java Web application runtime container based on Apache Tomcat 8.

Profile	Provides support for	Supported Java versions	Use
Java Web Tomcat 7 [page 863]	Some of the standard Java EE 6 APIs (Servlet, JSP, EL, Websocket)	7 (default); 8	**! Restriction** This runtime is deprecated. Support is discontinued as of 1 August 2020. We recommend using Java Web Tomcat 8. If you need a simplified Java Web application runtime container based on Apache Tomcat 7.
Java EE 6 Web Profile [page 863]	Java EE 6 Web Profile APIs	7 (default)	**! Restriction** This runtime is deprecated. Support is discontinued as of 1 August 2020. For more information, see Java EE 6 Web Profile [page 863]. If you need an application runtime container together with all containers defined by the Java EE 6 Web Profile specification.
Java Web [page 864]	Some of the standard Java EE 6 APIs (Servlet, JSP, EL, Websocket)	7 (default)	**! Restriction** This runtime is deprecated. Support is discontinued as of 31th December 2017. We recommend using Java Web Tomcat 8. For more information, see Java Web [page 864] (*Migrating to Java Web Tomcat* section). If you need a small standalone Java Web container.

For the complete list of supported APIs, see Supported Java APIs [page 868]

! Restriction

Support of Java 6 in the Neo environment is discontinued. You cannot deploy or start applications on Java 6.

- If you redeploy your application or deploy a new one, you will not be able to use Java 6 but you will have to use Java 7 instead.
- If you have a running application with Java 6, you have to redeploy it with Java 7.

→ Tip

We recommend that you restart your Java applications at least once every three months. This ensures you always use the latest updates available for your runtime, with the latest security and other fixes.

You can view an application's runtime status in your subaccount's application area in the cockpit (*Applications* > *Java Applications* > *Monitoring* > *Processes*).

- See Check the Process Status [page 1605] for more information how to check the runtime status

- See Start and Stop Applications [page 1602] for more information how to start and stop applications.

→ Tip

You can still run Java 6 complied applications with Java 7 as Java 7 backward compatible.

4.2.2.2.1 Java EE 7 Web Profile TomEE 7

The Java EE 7 Web Profile TomEE 7 provides implementation of the Java EE 7 Web Profile specification.

! Restriction

Java 7 for Java EE7 Web Profile TomEE7 runtime is deprecated. If you are running Java applications on that runtime with that Java version, migrate to Java 8 on the same runtime.

To do this, redeploy the applications or update their Java version. If you redeploy the applications, specify explicitly Java 8 as the version (see Deploying and Updating Applications [page 875]). In both cases, restart the applications afterwards (see restart command [page 1528]).

The current version of Java EE 7 Web Profile TomEE application runtime container (neo-javaee7-wp 1.x) provides implementation for the following Java Specification Requests (JSRs):

Specification version	JSR
Java EE 7 (Web Profile)	JSR - 342
Servlet 3.1	JSR - 340
JavaServer Pages (JSP) 2.3	JSR - 245
Expression Language (EL) 3.0	JSR - 341
Debugging Support for Other Languages 1.0	JSR - 45
Standard Tag Library for JavaServer Pages (JSTL) 1.2	JSR - 52
Java API for RESTful Web Services (JAX-RS) 2.0	JSR - 339
JavaServer Faces (JSF) 2.2	JSR - 344
Common Annotations for Java Platform 1.2	JSR - 250
Enterprise JavaBeans (EJB) Lite 3.2 i Note EJB Timer Servce is also supported (although not part of the EJB Lite specification).	JSR - 345
Java Transaction API (JTA) 1.2	JSR - 907
Java Persistence API (JPA) 2.1	JSR - 338
Dependency Injection for Java 1.0	JSR - 330

Specification version	JSR
Contexts and Dependency Injection for Java EE platform 1.1	JSR - 346
Bean Validation 1.1	JSR - 349
Managed Beans 1.0	JSR - 316
Interceptors 1.2	JSR - 345

Source: JSR-342 (Web Profile), Chapter 2: Web Profile Definition

For more information about the differences between EJB 3.2 and EJB 3.2 Lite, see the Java EE 7 specification, JSR 345: Enterprise JavaBeans, section 21.1.

Development Process

The Java EE 7 Web Profile TomEE 7 enables you to easily create your applications for SAP BTP.

For more information, see .

Migrating to Java EE 7 Web Profile TomEE 7 (from Another Runtime)

1. If required, change the application.
   In the general case, this step is **not** required.
   An exception are applications using the HTTP Destination API [page 136] (`com.sap.core.connectivity.api.http.HttpDestination`). In such case, you have to package `HttpDestination` in your application as library. Additionally, you may need to change the application code or destinations. For more information, see HttpDestination (Version 2) [page 149].
2. Redeploy (see Deploying and Updating Applications [page 875]) and restart the application afterwards (see restart command [page 1528] or Start and Stop Applications [page 1602]).
   If you have questions or problems, create a ticket in component *BC-NEO-RT-JAV*.

Related Information

Java EE at a Glance

Supported Java APIs [page 868]

4.2.2.2.2 Java Web Tomcat 9

Java Web Apache Tomcat 9 (Java Web Tomcat 9) is the next edition of the Java Web application runtime container that has all characteristics and features of its predecessor Java Web Tomcat 8.

This container leverages Apache Tomcat 9 Web container without modifications and also adds the already established set of SAP BTP services client APIs. Applications running in the Apache Tomcat 9 Web container are portable to Java Web Tomcat 9. Existing applications running in Java Web Tomcat 7 and Java Web Tomcat 8 application runtime containers can run unmodified in Java Web Tomcat 9 in case they share the same set of enabled APIs.

> **! Restriction**
>
> HTTP2 protocol is not supported at SAP BTP.

> **i Note**
>
> Applications that use HTTP Destination (version 1) API should adopt HTTP Destination (version 2) API.

The default supported Java version for Java Web Tomcat 9 is Java 8. Java 11 is also supported.

The current version of Java Web Tomcat 9 application runtime container (neo-java-web 4.x) provides implementation for the following set of Java Specification Requests (JSRs) defined specifications:

Specification version	JSR
Servlet 4.0	JSR - 369
JavaServer Pages (JSP) 2.3	JSR - 245
Expression Language (EL) 3.0	JSR - 341
Debugging Support for Other Languages 1.0	JSR - 45
Java API for WebSocket	JSR - 356

The following subset of APIs of SAP BTP services are available within Java Web Tomcat 9: document service APIs, mail service APIs, connectivity service APIs (destination configuration and authentication header provider), SAP HANA service and SAP ASE service JDBC APIs, and security APIs.

Migrating to Java Web Tomcat 9 (from Another Runtime)

1. Generally, you do not need to change the application. The only exception is the case if you are using the HTTP Destination API [page 136] (`com.sap.core.connectivity.api.http.HttpDestination`). In such case, you have to package `HttpDestination` in your application as library. Additionally, you may need to change the application code or destinations. For more information, see HttpDestination (Version 2) [page 149].
2. Redeploy (see Deploying and Updating Applications [page 875]) and restart the application afterwards (see restart command [page 1528] or Start and Stop Applications [page 1602]).
   If you have questions or problems, create a ticket in component *BC-NEO-RT-JAV*.

4.2.2.2.3 Java Web Tomcat 8

Java Web Apache Tomcat 8 (Java Web Tomcat 8) is the next edition of the Java Web application runtime container that has all characteristics and features of its predecessor Java Web Tomcat 7.

This container leverages Apache Tomcat 8.5 Web container without modifications and also adds the already established set of SAP BTP services client APIs. Applications running in the Apache Tomcat 8.5 Web container are portable to Java Web Tomcat 8. Existing applications running in Java Web and Java Web Tomcat 7 application runtime containers can run unmodified in Java Web Tomcat 8 in case they share the same set of enabled APIs.

> **!** Restriction
>
> HTTP2 protocol is not supported at SAP BTP.

The supported Java version for Java Web Tomcat 8 is Java 8.

The current version of Java Web Tomcat 8 application runtime container (neo-java-web 3.x) provides implementation for the following set of Java Specification Requests (JSRs) defined specifications:

Specification version	JSR
Servlet 3.1	JSR - 340
JavaServer Pages (JSP) 2.3	JSR - 245
Expression Language (EL) 3.0	JSR - 341
Debugging Support for Other Languages 1.0	JSR - 45
Java API for WebSocket	JSR - 356

The following subset of APIs of SAP BTP services are available within Java Web Tomcat 8: document service APIs, mail service APIs, connectivity service APIs (destination configuration and authentication header provider), SAP HANA service and SAP ASE service JDBC APIs, and security APIs.

Migrating to Java Web Tomcat 8 (from Another Runtime)

1. If required, change the application.
   In the general case, this step is **not** required.
   An exception are applications using the HTTP Destination API [page 136] (`com.sap.core.connectivity.api.http.HttpDestination`). In such case, you have to package `HttpDestination` in your application as library. Additionally, you may need to change the application code or destinations. For more information, see HttpDestination (Version 2) [page 149].
2. Redeploy (see Deploying and Updating Applications [page 875]) and restart the application afterwards (see restart command [page 1528] or Start and Stop Applications [page 1602]).
   If you have questions or problems, create a ticket in component *BC-NEO-RT-JAV*.

4.2.2.2.4 Java Web Tomcat 7

Java Web Tomcat 7 (deprecated runtime) is a simplified edition of Java Web application runtime container providing optimized performance particularly in the area of startup time and memory footprint.

> **i** Note
>
> As of 1 August 2020, we have discontinued the support for Java Web Tomcat 7 runtime. If you have applications running on it, migrate them to another runtime (we recommend Java Web Tomcat 8 [page 862]). For more information, see the migration steps below.

> **→** Tip
>
> Check if your applications use this runtime with the `status` console command (see status [page 1555]), or by exploring the application information in the cloud cockpit.

Migrating to Java Web Tomcat 8

Migrating to Java Web Tomcat 8 requires **no** changes to the application itself. Just redeploy (see Deploying and Updating Applications [page 875]) and restart the application afterwards (see restart command [page 1528] or Start and Stop Applications [page 1602]).

If you have questions or problems, create a ticket in component *BC-NEO-RT-JAV*.

4.2.2.2.5 Java EE 6 Web Profile

Deprecated. The Java EE 6 Web Profile application runtime container of SAP BTP is Java EE 6 Web Profile certified.

> **!** Restriction
>
> As of 1 August 2020, we have discontinued the support for Java EE 6 Web Profile runtime. If you are running Java applications on that runtime, migrate to another one as follows:
>
> - If you are using web profile features, migrate to Java EE 7 Web Profile TomEE 7 [page 859]
> - If you are not using those features, migrate to Java Web Tomcat 8 [page 862].
>
> For more information, see the migration steps below.

> **→** Tip
>
> Check if your applications use this runtime with the `status` console command (see status [page 1555]), or by exploring the application information in the cloud cockpit.

Migrating to Java Web Tomcat 8 or Java EE 7 Web Profile TomEE 7

1. If required, change the application.
   In the general case, this step is **not** required.
   An exception are applications using the HTTP Destination API [page 136] (`com.sap.core.connectivity.api.http.HttpDestination`). In such case, you have to package

`HttpDestination` in your application as library. Additionally, you may need to change the application code or destinations. For more information, see HttpDestination (Version 2) [page 149].

2. Redeploy (see Deploying and Updating Applications [page 875]) and restart the application afterwards (see restart command [page 1528] or Start and Stop Applications [page 1602]).
   If you have questions or problems, create a ticket in component *BC-NEO-RT-JAV*.

> → Tip
>
> You can check the application runtime by using the `status` console command (see status [page 1555]) or by exploring the application information in the cloud cockpit.

4.2.2.2.6 Java Web

Java Web is a minimalistic application runtime container in SAP BTP that offers a subset of Java EE standard APIs typical for a standalone Java Web Container.

> ! Restriction
>
> This runtime is deprecated. We recommend migrating to Java Web Tomcat 8. For more information, see below.
>
> If you have concerns, please create a ticket in component *BC-NEO-RT-JAV*.

Migrating to Java Web Tomcat 8

1. If required, change the application.
   In the general case, this step is **not** required.
   An exception are applications using the HTTP Destination API [page 136] (`com.sap.core.connectivity.api.http.HttpDestination`). In such case, you have to package `HttpDestination` in your application as library. Additionally, you may need to change the application code or destinations. For more information, see HttpDestination (Version 2) [page 149].
2. Redeploy (see Deploying and Updating Applications [page 875]) and restart the application afterwards (see restart command [page 1528] or Start and Stop Applications [page 1602]).
   If you have questions or problems, create a ticket in component *BC-NEO-RT-JAV*.

Related Information

Choose JRE Version [page 1595]
Supported Java APIs [page 868]

4.2.2.2.7 Migrating from an Unsupported Runtime

Migrate an application running on an unsupported Java runtime (such as Java Web, Java EE 6 Web Profile or Java Web Tomcat 7) to a supported one (Java EE 7 Web Profile TomEE 7 or Java Web Tomcat 8).

Overview

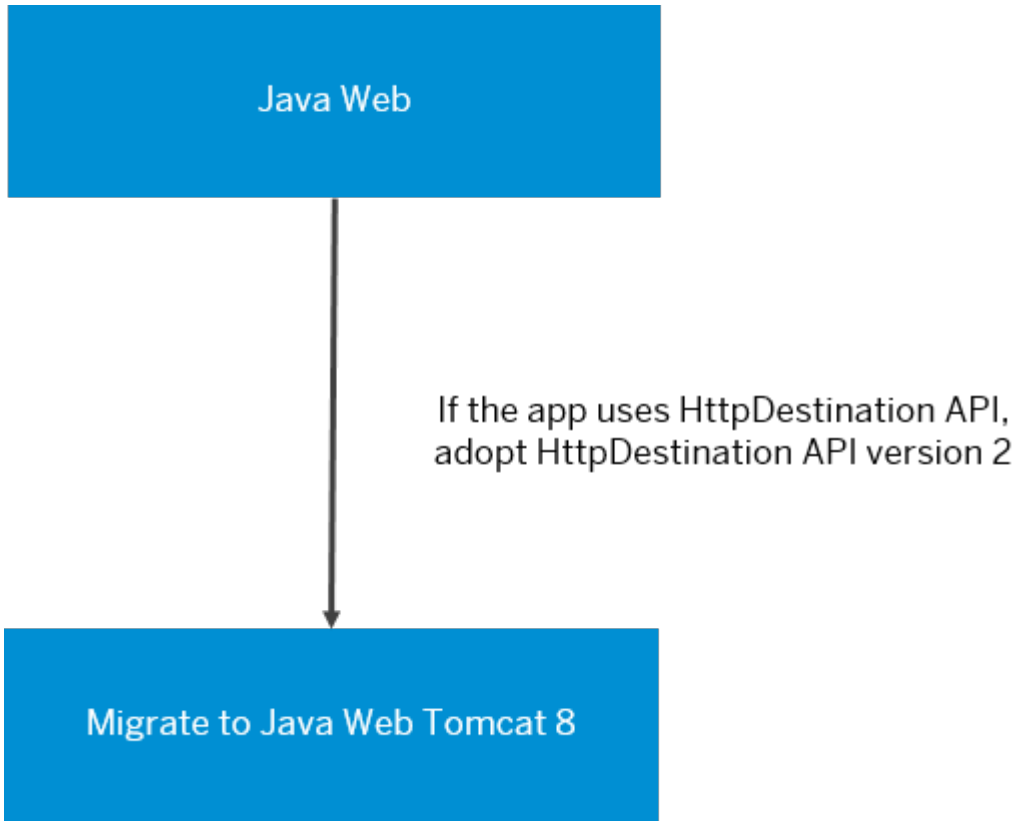
> → Tip
>
> If you no longer need this application, skip the migration and directly stop (if started) and undeploy the application (see Start and Stop Applications [page 1602] or stop [page 1562] / undeploy [page 1574] command).

If you still need the application, then you have the following migration paths:

Java Web

If your application runs on Java Web, migrate it to **Java Web Tomcat 8 runtime with Java 8**.

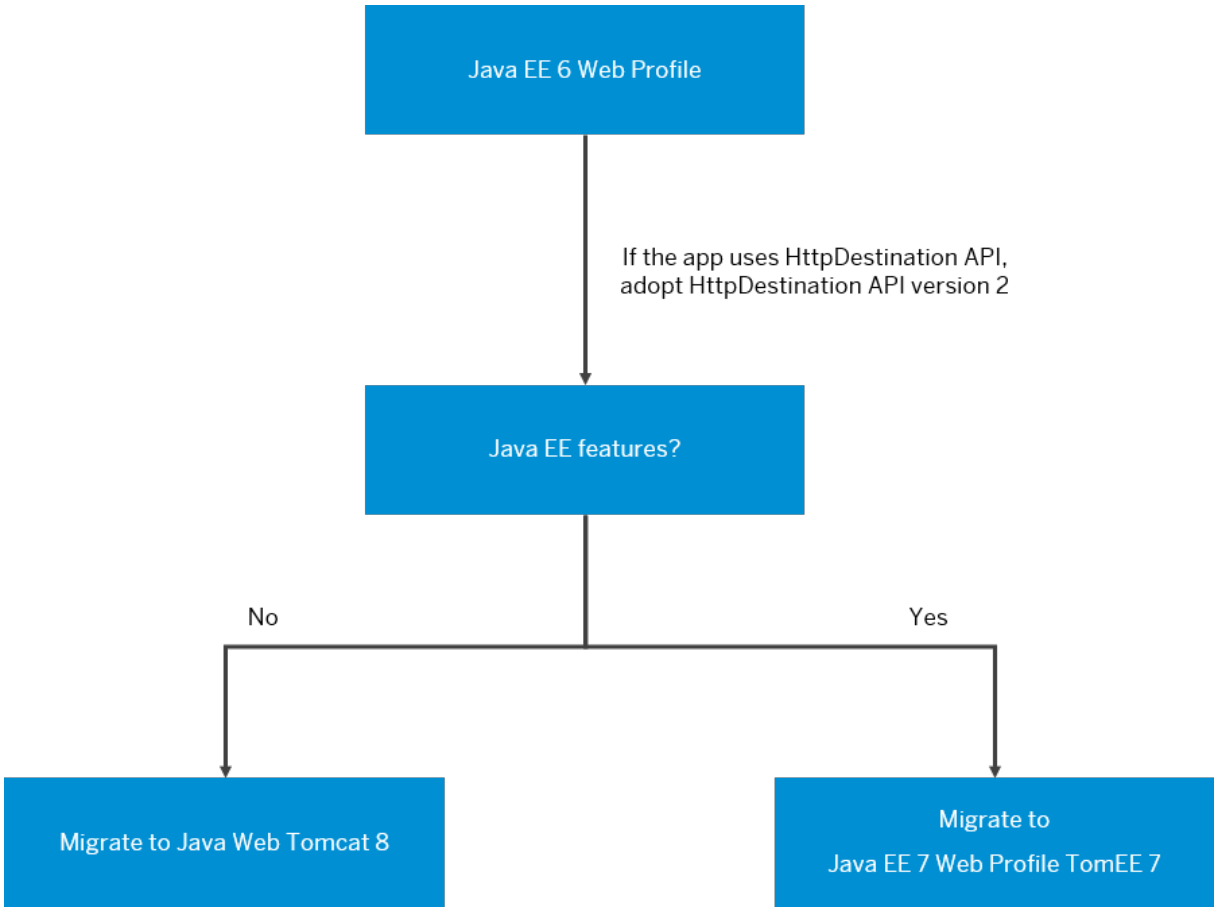
Before migrating, check if the application uses HttpDestination API. If it does, switch to HttpDestination API (version 2) (this is the version of the API available in the new runtime).



Java EE 6 Web Profile

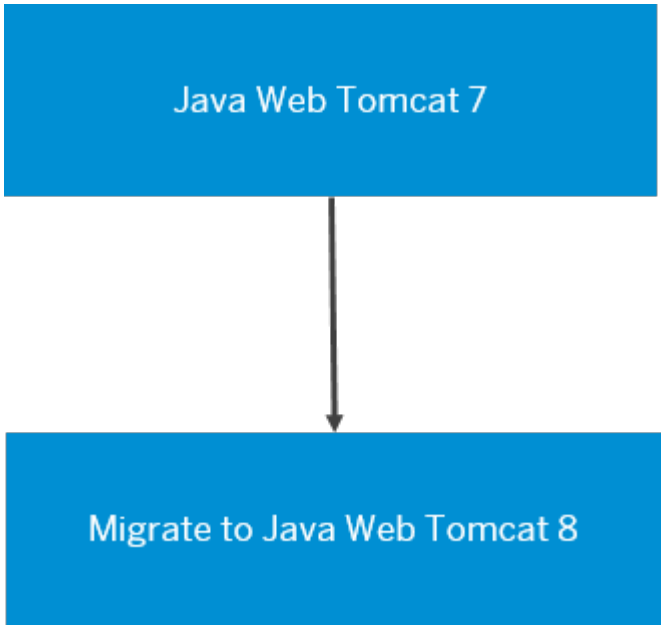
- Check if the application uses HttpDestination API. If it does, switch to HttpDestination API (version 2) (this is the version of the API available in the new runtime).

- Check if the application is using Java EE features, different from servlets, jsp, el, websockets (e.g., EJBs, CDI, container managed transactions, etc.):
  - If it does not use such features, migrate to **Java Web Tomcat 8 runtime with Java 8**.
  - If it uses such features, migrate to **Java EE 7 Web Profile TomEE 7** runtime.



Java Web Tomcat 7

Migrate it to **Java Web Tomcat 8 runtime with Java 8**.



Migration Steps

1. Redeploy it (see Deploying and Updating Applications [page 875].)
2. Restart it (see Start and Stop Applications [page 1602] or restart [page 1528] console command).

> → Tip
>
> You may also consider Update Applications with Zero Downtime [page 1610] or rolling-update [page 1534], both of which are performed only from command line.

4.2.2.3 Compute Units

A compute unit is the virtualized hardware resources used by an SAP BTP application.

After being deployed to the cloud, the application is hosted on a compute unit with certain central processing unit (CPU), main memory, disk space, and an installed OS.

Compute Unit Sizes

SAP BTP offers four standard sizes of compute units according to the provided resources.

Depending on their needs, customers can choose from the following compute unit configurations:

Compute Unit	Configuration	Size Parameter Value
Lite edition	1 CPU Core; 3072 MB Main memory	*lite*
Professional edition	2 CPU Cores; 4096 MB Main memory	*pro*
Premium edition	4 CPU Cores; 8192 MB Main memory	*prem*
Premium Plus edition	8 CPU Cores; 16384 MB Main memory	*prem-plus*

The third column in the table shows what value of the `-z` or `--size` parameter you need to use for a console command.

For customer accounts, all sizes of compute units are available. During deployment, customers can specify the compute unit on which they want their application to run.

For more information, see deploy [page 1422].

Related Information

Purchase a Customer Account
Account Model
Configure Entitlements and Quotas for Subaccounts

4.2.3 Development Environment

The basic tools of the SAP BTP development environment, the SAP BTP Tools, include the SAP BTP SDK for Neo environment.

The SAP BTP SDK for Neo environment, on the other hand, contains everything you need to work with the platform, including a local server runtime and a set of command line tools. The command line capabilities allow modern build tools, such as Apache Maven, to be used to professionally produce Web applications for the cloud. The command line is particularly important for setting up and automating a headless continuous build and test process.

Related Information

Setting Up the Development Environment [page 845]

4.2.3.1 Supported Java APIs

When you develop applications that run on SAP BTP, you can rely on certain Java EE standard APIs. These APIs are provided with the runtime of the platform. They are based on standards and are backward compatible as defined in the Java EE specifications. Currently, you can make use of the APIs listed below:

- javax.activation
- javax.annotation
- javax.el
- javax.mail
- javax.persistence
- javax.servlet
- javax.servlet.jsp
- javax.servlet.jsp.jstl
- javax.websocket

You can also make use of the following third-party APIs:

- org.slf4j.Logger
- org.slf4j.LoggerFactory

If you are using the SAP BTP SDK for Java EE 6 WebProfile, you can have access to the following Java EE APIs as well:

- javax.faces
- javax.validation
- javax.inject
- javax.ejb
- javax.interceptor
- javax.transaction
- javax.enterprise
- javax.decorator

The table below summarizes the Java Request Specifications (JSRs) supported in the two SAP BTP SDKs for Java.

Supported Java EE 6 Specification	SDK for Java Web	SDK for Java EE 6 WebProfile
Servlet 3.0	yes	yes
JavaServer Pages (JSP) 2.2	yes	yes
Expression Language (EL) 2.2	yes	yes
Debugging Support for Other Languages 1.0	yes	yes
Standard Tag Library for JavaServer Pages (JSTL) 1.2	yes	yes
Common Annotations for Java Platform 1.1	yes	yes
Java API for WebSocket	yes	yes
JavaServer Faces (JSF) 2.0	no	yes
Enterprise JavaBeans (EJB) Lite 3.1	no	yes

Supported Java EE 6 Specification	SDK for Java Web	SDK for Java EE 6 WebProfile
Java Transaction API (JTA) 1.1	no	yes
Java Persistence API (JPA) 2.0	no	yes
Dependency Injection for Java 1.0	no	yes
Contexts and Dependency Injection for Java EE platform 1.0	no	yes
Bean Validation 1.0	no	yes
Managed Beans 1.0	no	yes
Interceptors 1.1	no	yes

The table below summarizes the Java Request Specifications (JSRs) supported in the SAP BTP SDK for Java Web Tomcat 8 .

Supported Java EE 7 Specification	SDK for Java Web Tomcat 8	SDK for Java EE 7 Web Profile TomEE 7
Servlet 3.1	yes	yes
JavaServer Pages (JSP) 2.3	yes	yes
Expression Language (EL) 3.0	yes	yes
WebSocket 1.1	yes	yes
Common Annotations for Java Platform 1.2	yes	yes
Standard Tag Library for JavaServer Pages (JSTL) 1.2	yes	yes
Debugging Support for Other Languages 1.0	yes	yes
Java API for RESTful Web Services (JAX-RS) 2.0	no	yes
JavaServer Faces (JSF) 2.2	no	yes
Enterprise JavaBeans (EJB) Lite 3.2	no	yes
Java Transaction API (JTA) 1.2	no	yes
Java Persistence API (JPA) 2.1	no	yes
Dependency Injection for Java 1.0	no	yes
Contexts and Dependency Injection for Java EE platform 1.1	no	yes
Bean Validation 1.1	no	yes

Supported Java EE 7 Specification	SDK for Java Web Tomcat 8	SDK for Java EE 7 Web Profile TomEE 7
Managed Beans 1.0	no	yes
Interceptors 1.2	no	yes

In addition to the standard APIs, SAP BTP offers platform-specific services that define their own APIs that can be used from the SAP BTP SDK. The APIs of the platform-specific services are listed in the table below

API	More Information
Security	Securing Java Applications [page 1676]
Connectivity	Connectivity and Destination APIs [page 135]
Document	Handling CMIS Metadata [page 570]

The SAP BTP SDK contains a platform API folder for compiling your Web applications. It contains the above content, that is, all standard and third-party API JARs (for legal reasons provided "as is", that is, they also have non-API content on which you should not rely) and the platform APIs of the SAP BTP services.

You can add additional (pure Java) application programming frameworks or libraries and use them in your applications. For example, you can include Spring Framework in the application (in its application archive) and use it in the application. In such cases, the application should handle all dependencies to such additional frameworks or libraries and you should take care for the whole assembly of such additional frameworks or libraries inside the application itself.

SAP BTP also provides numerous other capabilities and APIs that might be accessible for applications. However, you should rely only on the APIs listed above.

For more information, see API Documentation [page 1147].

Related Information

Java Web [page 864]
Java EE 6 Web Profile [page 863]
Java Web Tomcat 7 [page 863]
Java Web Tomcat 8 [page 862]
Java EE 7 Web Profile TomEE 7 [page 859]

4.2.4 Developing Java Applications

You can develop applications for SAP BTP just like for any application server. Inside, you can embed the usage of the services provided by the platform.

Development Environment

SAP BTP development environment is designed and built to optimize the process of development and deployment.

For Java applications, you can choose between three types of SAP BTP SDK for Neo environment:

- SDK for Java Web - provides support for some of the standard Java EE 6 APIs (Servlet, JSP, EL, Websocket)
- SDK for Java Web Tomcat 7 - provides support for some of the standard Java EE 6 APIs (Servlet, JSP, EL, Websocket)
- SDK for Java EE 6 Web Profile - certified to support Java EE 6 Web Profile APIs
- SDK for Java Web Tomcat 8 - provides support for some of the standard Java EE 7 APIs (Servlet, JSP, EL, Websocket)

For more information, see Development Environment [page 868]

Use Java EE 6 Web Profile

SAP BTP is Java EE 6 Web Profile certified so you can extend the basic functionality of your application with Java EE 6 Web Profile technologies. If you are working with the SDK for Java EE 6 Web Profile, you can equip the basic application with additional Java EE features, such as EJB, CDI, JTA.

Use Services and Utilities

Create a fully-fledged application benefiting from the capabilities and services provided by SAP BTP. In your application, you can choose to use:

- Authentication [page 1676] - by default, SAP BTP is configured to use SAP ID service as identity provider (IdP), as specified in SAML 2.0. You can configure trust to your custom IdP, to provide access to the cloud using your own user database.
- UI development toolkit for HTML5 (SAPUI5) - use the platform's official UI framework.
- Connectivity Service [page 134] - use it to connect Web applications to Internet, make on-demand to on-premise connections to Java and ABAP on-premise systems and configure destinations to send and fetch e-mail.
- Logging - implement a logging API if you want to have logs produced at runtime.
- Cloud Environment Variables [page 873]- use system environment variables that identify the runtime environment of the application.

Deploy

First, deploy and test the ready application on the local runtime and then make it available on SAP BTP.

For more information, see Deploying and Updating Applications [page 875]

You can speed up your development by applying and activating new changes on the already running application. Use the hot-update command.

For more information, see hot-update [page 1473]

Manage

Manage all applications deployed in your account from a single dedicated user interface - SAP BTP cockpit.

Monitor

Configure checks and monitor the state of your applications.

For more information, see Java: Application Operations [page 1589]

4.2.4.1 Using Cloud Environment Variables

Overview

SAP BTP runtime sets several system environment variables that identify the runtime environment of the application. Using them, an application can get information about its application name, subaccount and URL, as well as information about the region host it is deployed on and region-specific parameters. All SAP BTP specific environment variables names start with the common prefix HC_.

List of Environment Variables

The following SAP BTP environment variables are set to the runtime environment of the application:

Key	Sample Value	Description
HC_HOST	`hana.ondemand.com` / `us1.hana.ondemand.com` / `hanatrial.ondemand.com`	Base URL of the SAP BTP region host where the application is deployed
HC_HOST_SVC	`svc.hana.ondemand.com` / `svc.us1.hana.ondemand.com` / `svc.hanatrial.ondemand.com`	URL of the SAP BTP region host which provides access within the same region; for direct communication and not open on the Internet or other networks

Key	Sample Value	Description
HC_HOST_CERT	**cert.hana.ondemand.com / cert.us1.hana.ondemand.com / cert.hanatrial.ondemand.com**	URL of the SAP BTP region host which enables client certificate authentication
HC_REGION	**EU_1** / **US_1**	Region where the application is deployed
HC_ACCOUNT	**mysubaccount**	Name of the subaccount where the application is deployed
HC_APPLICATION	**myapp**	Application name
HC_APPLICATION_URL	**https://myapp.hana.ondemand.com**	URL of the application
HC_LOCAL_HTTP_PORT	**9001**	HTTP port of the application bound to localhost
HC_LANDSCAPE	**production** / **trial**	Type of the region host where the application is deployed
HC_PROCESS_ID	**8921b0a7cebc5538038b6b7b0c0ea6a7127f0cd4**	Process ID of the current application process as returned by the `status` command with parameter `--show-full-process-id`.
HC_AVAILABILITY_ZONE	**AZ 1** / **AZ 2**	Name of the availability zone where the application process is running
HC_OP_HTTP_PROXY_HOST	**localhost**	Host of the HTTP Proxy for on-premise connectivity
HC_OP_HTTP_PROXY_PORT	**20003**	Port of the HTTP Proxy for on-premise connectivity

Using Cloud Environment Variables

SAP BTP environment variables are accessed as standard system environment variables of the Java process - for example via System.getenv("...").

> **i** Note
>
> Environment variables are not set when deploying locally with the console client.

Example

```
<html>
  <head>
    <title>Display SAP BTP Environment Platform variables</title>
```

```
  </head>
  <body>
        <p>Application Name: <%= System.getenv("HC_APPLICATION") %></p>
  </body>
</html>
```

Related Information

status [page 1555]
Regions
HTTP Proxy for On-Premise Connectivity [page 152]

4.2.5 Deploying and Updating Applications

Content
Deploying Applications [page 875]
Updating Application Properties [page 876]
Updating During Development [page 876]
Updating Productive Applications [page 876]

Deploying Applications

After you have created your Java application, you need to deploy and run it on SAP BTP. We recommend that you first deploy and test your application on the local runtime before deploying it on the cloud. Use the tool that best fits your scenario:

Console Client	Deploy Locally with the Console Client [page 890] Deploy on the Cloud with the Console Client [page 891]	You want to deploy an application in the form of one or more WAR files. Command: `deploy`
Cockpit	Deploy on the Cloud with the Cockpit [page 894]	You want to deploy an application in the form of a WAR file.
Lifecycle Management API	Deploy an Application [page 878]	You want to deploy an application in the form of one or more WAR files.

Updating Application Properties

Application properties are configured during deployment with a set of parameters. To update these properties, use one of the following approaches:

Console Client	deploy [page 1422]	Deploy the application with new WAR file(s) and make changes to the configuration parameters. Command: `deploy`
Console Client	set-application-property [page 1541]	Change some of the application properties you defined during deployment without redeploying the application binaries. Command: `set-application-property`
Cockpit	Deploy on the Cloud with the Cockpit [page 894]	Update the application with a new WAR file or make changes to the configuration parameters.

Updating During Development

If you want to quickly see your changes while developing an application, use the following approaches:

Console Client	hot-update [page 1473]	Apply and activate changes. Use the command to speed up development and not for updating productive applications. Command: `hot-update`
Console Client	Use Delta Deployment [page 893] deploy [page 1422] hot-update [page 1473]	Apply changes in a deployed application without uploading the entire set of files. Command: `deploy` or `hot-update` Parameter: `--delta`

Updating Productive Applications

If you are an application operator and need to deploy a new version of a productive application or perform maintenance, you can choose among several approaches:

Zero Downtime	Update Applications with Zero Downtime [page 1610] rolling-update [page 1534]	Use when the new application version is backward compatible with the old version. Deploy a new version of the application and disable and enable processes in a rolling manner, or, do it at one go with the `rolling-update` command.

Planned Downtime **(Maintenance Mode)**	Enable Maintenance Mode for Planned Downtimes [page 1613]	Use when the new application version is backward incompatible. Enable maintenance mode for the time of the planned downtime.
Soft Shutdown	Perform Soft Shutdown [page 1615]	Supports zero downtime and planned downtime scenarios. Disable the application or individual processes in order to shut down the application or processes gracefully.

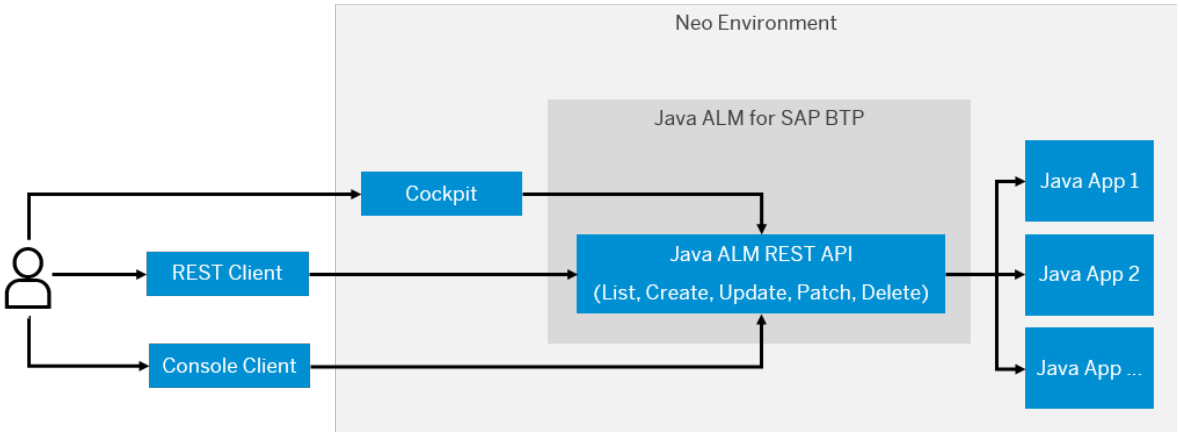
Related Information

Prerequisites and Restrictions

4.2.5.1 Java ALM API Tutorial

The Java ALM service REST API provides functionality for managing the lifecycle of Java applications.

Overview



This tutorial provides information about the most common use cases for Java applications and the operations that are included in each one:

- Authentication and Authorization [page 878]
- Deploy an Application [page 878]
- Deploy an Application with a Copy Operation [page 882]
- Verify Application Binaries [page 885]
- Start an Application [page 887]

- Stop an Application [page 889]

4.2.5.1.1 Authentication and Authorization

You can use one of the following:

- Basic authentication
  You provide username and password.
- OAuth authentication and authorization
  The REST API is protected with OAuth 2.0 client credentials.

Prerequisites

- For basic authentication, assign the *manageJavaApplications* scope to the platform role used in the subaccount. See Platform Scopes [page 1307].
- For OAuth authentication and authorization, create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149] as you add the *Lifecycle Management* scopes for the *Platform API OAuth* client.

4.2.5.1.2 Deploy an Application

Prerequisites

- You have installed a REST client.

Context

For the purposes of this tutorial, we will deploy three `.war` files: (app.war, example.war, demo.war).

> **i** Note
>
> The deployment with an unsupported runtime will fail with an error message, and the deployment with a deprecated runtime will result in a warning message.

Procedure

1. Get a CSRF token.

a. Start a new session.
b. Send a *GET* CSRF Protection request:

```
Client Request:
GET https: api.hana.ondemand.com/lifecycle/v1/csrf
Request Headers:
X-CSRF-Token: Fetch
Authorization: Basic UDE5NDE3OTM5NDg6RnJhZ28jNjQ3Ng==
```

For OAuth Platform API authentication and authorization, the last line looks like `Authorization: Bearer a9cd683534471f499b630bb97b3d3fc`, where *a9cd683534471f499b630bb97b3d3fc* has been retrieved with `POST <host>/oauth2/apitoken/v1?grant_type=client_credentials`. For more information, see Using Platform APIs [page 1149].

```
Server Response:
Response Status: 200
Response Headers:
X-CSRF-Token: 3B95B7A8B0E8E6B923C67E6C0BFD234D
```

> **i Note**
>
> After a while the CSRF token expires. If you are using an invalid CSRF token, you will receive an error message similar to this one: `HTTP Status 403 - CSRF token validation failed!` If this happens, get a new token.

2. Create an application.

   Send a *POST* Applications request:

```
Client Request:
POST: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps
Request Body:
{
                "applicationName": "myapp",
                "runtimeName": "neo-java-web",
                "runtimeVersion": "1",
                "minProcesses": 1,
                "maxProcesses": 1
}
```

```
Server Response:
Response Status: 201
Response Body:
{
                "metadata": {
                                "url": "/lifecycle/v1/accounts/test/apps/myapp"
                },
                "entity": {
                                "accountName": "test",
                                "applicationName": "myapp",
                                "runtimeName": "neo-java-web",
                                "runtimeVersion": "1",
                                "minProcesses": 1,
                                "maxProcesses": 1
                }
}
```

> ### → Tip
>
> You can add other properties to the body of the request. The properties in this example are the minimum requirements that let you execute the request successfully.

3. Create a list of the application binaries.

   Send a *POST* Binaries request:

```
Client Request:
POST: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps/myapp/
binaries
Request Headers:
X-CSRF-Token: 3B95B7A8B0E8E6B923C67E6C0BFD234D
Content-Type: application/json
Request Body:
{
               "files": [{
                              "path": "app.war"
               }, {
                              "path": "demo.war"
               }, {
                              "path": "example.war"
               }]
}
```

```
Server Response:
Response Status: 201
Response Body:
{
               "metadata": {
                              "url": "/lifecycle/v1/accounts/test/apps/myapp/
binaries"
               },
               "entity": {
                              "totalSize": 0,
                              "status": "UPLOADING",
                              "files": [{
                                             "path": "app.war",
                                             "pathGuid": "YXBwLndhcg\u003d
\u003d",
                                             "status": "UNAVAILABLE",
                                             "entries": []
                              }, {
                                             "path": "example.war",
                                             "pathGuid": "ZXhhbXBsZS53YXI
\u003d",
                                             "status": "UNAVAILABLE",
                                             "entries": []
                              }, {
                                             "path": "demo.war",
                                             "pathGuid": "ZGVtby53YXI\u003d",
                                             "status": "UNAVAILABLE",
                                             "entries": []
                              }]
               }
}
```

   This request describes the metadata of the binaries and prepares them for their upload.

4. Upload the binaries.

> ### Note
>
> You must start uploading the binaries within the next 2 minutes. Otherwise, the operation will be canceled and you will have to deploy the application again. If you do not start uploading the binaries within the next 2 minutes, you will receive the following response:

```
Server Response:
Response Status: 404
Response Body:
{
    "code": "98a59939-0e9a-430c-9ec3-c094a4d8d78d",
    "description": "Application operation is not found"
}
```

Send *PUT* Binary requests for each one of the binaries. Use the corresponding *pathGuid* values for each `.war` file from the previous *POST* Binaries response and add it to the URL:

```
Client Request:
PUT: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps/myapp/
binaries/YXBwLndhcg==
Request Headers:
X-CSRF-Token: 3B95B7A8B0E8E6B923C67E6C0BFD234D
Content-Type: application/octet-stream
Request Body:
Choose to add a file and select app.war.
```

```
Client Request:
PUT: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps/myapp/
binaries/ZXhhbXBsZS53YXI=
Request Headers:
X-CSRF-Token: 3B95B7A8B0E8E6B923C67E6C0BFD234D
Content-Type: application/octet-stream
Request Body:
Choose to add a file and select example.war.
```

```
Client Request:
PUT: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps/myapp/
binaries/ZGVtby53YXI=
Request Headers:
X-CSRF-Token: 3B95B7A8B0E8E6B923C67E6C0BFD234D
Content-Type: application/octet-stream
Request Body:
Choose to add a file and select demo.war.
```

If the operation is successful, the response for all three requests should return 200 without a body.

5. List the binaries.

   Send a *GET* Binaries request every 5-10 seconds:

   ```
   Client Request:
   GET: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps/myapp/
   binaries
   ```

   Repeat the process and observe the general status until you receive the *FAILED* or the *DEPLOYED* values. The *DEPLOYED* status shows that the deployment operation has been successful and you can now start your application:

   ```
   Server Response:
   Response Status: 200
   Response Body:
   ```

```
{
               "metadata": {},
               "entity": {
                              "totalSize": 57857,
                              "status": "DEPLOYED",
                              "warnings": "Warning: No compute unit size was
specified for the application so size was set automatically to \u0027lite
\u0027.",
                              "files": [{
                                             "path": "app.war",
                                             "pathGuid": "YXBwLndhcg\u003d
\u003d",
                                             "size": 17194,
                                             "status": "AVAILABLE",
                                             "hash":
"6c8b99a72d5b42db31cc576273260f9c2f316c1ac7dcc4a8c845412e51d420f0dcf53f4035745
e303cdd43bf73974fada19839920d845010013bf422ae5bc4dd",
                                             "entries": [{...}]
                              }, {
                                             "path": "example.war",
                                             "pathGuid": "ZXhhbXBsZS53YXI
\u003d",
                                             "size": 37615,
                                             "status": "AVAILABLE",
                                             "hash":
"7b2a80771f79d0740f629bdaaf019c550b10df55eec8789447ec02fa93e7fdb1f6f47f4864769
f4a4f027a4bca8bfa1ea45a83c5fb38ae539b397abe9fe66be1",
                                             "entries": [{...}]
                              }, {
                                             "path": "demo.war",
                                             "pathGuid": "ZGVtby53YXI\u003d",
                                             "size": 3048,
                                             "status": "AVAILABLE",
                                             "hash":
"8c4b39bfe3a034d64e8592e7cf638ac4b5985c5f9a4f691270d040b8f15dc8edbb6284bd5431f
1a240abaad3b2288411563b784b691c35ca677ae5e9ced565a9",
                                             "entries": [{...}]
                              }]
               }
}
```


The binaries are now officially *DEPLOYED*. You can also see that each binary has status *AVAILABLE*.

4.2.5.1.3 Deploy an Application with a Copy Operation

You can deploy an application by copying an existing application.

Prerequisites

- For basic authentication, assign the *Administrator* role to the user.
- You have an existing source application deployed on the platform.
- You have installed a REST client.

Context

In this tutorial, you will deploy an application from an existing application by specifying the source account and application as query parameters.

> **i Note**
>
> In platform OAuth, the copy operation is applicable for applications in the same account.

Procedure

1. Get a CSRF token.
   a. Start a new session.
   b. Send a *GET* CSRF Protection request:

```
Client Request:
GET https: api.hana.ondemand.com/lifecycle/v1/csrf
Request Headers:
X-CSRF-Token: Fetch
Authorization: Basic UDE5NDE3OTM5NDg6RnJhZ28jNjQ3Ng==
```

   For OAuth Platform API authentication and authorization, the last line looks like `Authorization: Bearer a9cd683534471f499b630bb97b3d3fc`, where *a9cd683534471f499b630bb97b3d3fc* has been retrieved with `POST <host>/oauth2/apitoken/v1?grant_type=client_credentials`. For more information, see Using Platform APIs [page 1149].

```
Server Response:
Response Status: 200
Response Headers:
X-CSRF-Token: 3B95B7A8B0E8E6B923C67E6C0BFD234D
```

   > **i Note**
   >
   > After a while the CSRF token expires. If you are using an invalid CSRF token, you will receive an error message similar to this one: `HTTP Status 403 - CSRF token validation failed!` If this happens, get a new token.

2. Create an application with a copy operation of the source application.

   Send a *POST* Applications request:

```
Client Request:
POST: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps?
operation=copy&sourceAccount=sourcesubaccount&sourceApplication=sourceapp
Request Body:
{
                "applicationName": "myapp",
                "runtimeName": "neo-java-web",
                "runtimeVersion": "1",
                "minProcesses": 1,
                "maxProcesses": 1
}
```

```
Server Response:
```

```
Response Status: 201
Response Body:
{
                "metadata": {
                                "url": "/lifecycle/v1/accounts/test/apps/myapp"
                },
                "entity": {
                                "accountName": "test",
                                "applicationName": "myapp",
                                "runtimeName": "neo-java-web",
                                "runtimeVersion": "1",
                                "minProcesses": 1,
                                "maxProcesses": 1
                }
}
```

> **→ Tip**
>
> The body is optional for the request. If you do not specify a body, the REST API will take the parameters from the source application.

The application is deployed by using the binaries in the source application.

3. List the binaries.

   Send a *GET* Binaries request every 5-10 seconds:

```
Client Request:
GET: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps/myapp/
binaries
```

   Repeat the process and observe the general status until you receive the *FAILED* or the *DEPLOYED* values. The *DEPLOYED* status shows that the copy operation has been successful and you can now start your application:

```
Server Response:
Response Status: 200
Response Body:
{
                "metadata": {},
                "entity": {
                                "totalSize": 57857,
                                "status": "DEPLOYED",
                                "warnings": "Warning: No compute unit size was
specified for the application so size was set automatically to \u0027lite
\u0027.",
                                "files": [{
                                                "path": "app.war",
                                                "pathGuid": "YXBwLndhcg\u003d
\u003d",
                                                "size": 17194,
                                                "status": "AVAILABLE",
                                                "hash":
"6c8b99a72d5b42db31cc576273260f9c2f316c1ac7dcc4a8c845412e51d420f0dcf53f4035745
e303cdd43bf73974fada19839920d845010013bf422ae5bc4dd",
                                                "entries": [{...}]
                                }, {
                                                "path": "example.war",
                                                "pathGuid": "ZXhhbXBsZS53YXI
\u003d",
                                                "size": 37615,
                                                "status": "AVAILABLE",
                                                "hash":
"7b2a80771f79d0740f629bdaaf019c550b10df55eec8789447ec02fa93e7fdb1f6f47f4864769
f4a4f027a4bca8bfa1ea45a83c5fb38ae539b397abe9fe66be1",
```

```
                                                "entries": [{...}]
                                    }, {
                                                "path": "demo.war",
                                                "pathGuid": "ZGVtby53YXI\u003d",
                                                "size": 3048,
                                                "status": "AVAILABLE",
                                                "hash":
"8c4b39bfe3a034d64e8592e7cf638ac4b5985c5f9a4f691270d040b8f15dc8edbb6284bd5431f
1a240abaad3b2288411563b784b691c35ca677ae5e9ced565a9",
                                                "entries": [{...}]
                                    }]
                        }
}
```

The binaries are now officially *DEPLOYED*. You can also see that each binary has status *AVAILABLE*.

4.2.5.1.4 Verify Application Binaries

Use the response to verify the binaries of an application.

Prerequisites

You have installed a REST client.

Context

You can validate the content of an application by verifying the hash values in a binaries response. For example, you verify changes to an application by comparing hash values of deployed binaries with the hash values of modified binaries. You can use this verification to be sure that you have the correct binaries for deploy or update in a copy operation.

Procedure

1. Get a CSRF token. If you try to start your application long after its deployment, the token has most probably expired.

   Send a *GET* CSRF Protection request.

   > **i Note**
   >
   > If your session is still actively running, you do not have to request a new CSRF token. In this case, we will use the CSRF token generated during the deployment scenario.

2. List the binaries.

Send a *GET* Binaries request every 5-10 seconds:

```
Client Request:
GET: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps/myapp/
binaries
```

Repeat the process and observe the general status until you receive the *FAILED* or the *DEPLOYED* values. The *DEPLOYED* status shows that the deployment operation has been successful and you can now start your application:

```
Server Response:
Response Status: 200
Response Body:
{
               "metadata": {},
               "entity": {
                              "totalSize": 57857,
                              "status": "DEPLOYED",
                              "warnings": "Warning: No compute unit size was
specified for the application so size was set automatically to \u0027lite
\u0027.",
                              "files": [{
                                             "path": "app.war",
                                             "pathGuid": "YXBwLndhcg\u003d
\u003d",
                                             "size": 17194,
                                             "status": "AVAILABLE",
                                             "hash":
"6c8b99a72d5b42db31cc576273260f9c2f316c1ac7dcc4a8c845412e51d420f0dcf53f4035745
e303cdd43bf73974fada19839920d845010013bf422ae5bc4dd",
                                             "entries": [{...}]
                              }, {
                                             "path": "example.war",
                                             "pathGuid": "ZXhhbXBsZS53YXI
\u003d",
                                             "size": 37615,
                                             "status": "AVAILABLE",
                                             "hash":
"7b2a80771f79d0740f629bdaaf019c550b10df55eec8789447ec02fa93e7fdb1f6f47f4864769
f4a4f027a4bca8bfa1ea45a83c5fb38ae539b397abe9fe66be1",
                                             "entries": [{...}]
                              }, {
                                             "path": "demo.war",
                                             "pathGuid": "ZGVtby53YXI\u003d",
                                             "size": 3048,
                                             "status": "AVAILABLE",
                                             "hash":
"8c4b39bfe3a034d64e8592e7cf638ac4b5985c5f9a4f691270d040b8f15dc8edbb6284bd5431f
1a240abaad3b2288411563b784b691c35ca677ae5e9ced565a9",
                                             "entries": [{...}]
                              }]
               }
}
```

The binaries are now officially *DEPLOYED*. You can also see that each binary has status *AVAILABLE*.

3. Use the hash values of the binaries to compare with those of previous binaries before you start another operation.

4.2.5.1.5 Start an Application

Procedure

1. Get a CSRF token. If you try to start your application long after its deployment, the token has most probably expired.

   Send a *GET* CSRF Protection request.

   > **i** Note
   >
   > If your session is still actively running, you do not have to request a new CSRF token. In this case, we will use the CSRF token generated during the deployment scenario.

2. Start the previously deployed application.

   Send a *PUT* Application State request and set the value of the *applicationState* property to *STARTED*:

```
Client Request:
PUT: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps/myapp/state
Request Headers:
X-CSRF-Token: 3B95B7A8B0E8E6B923C67E6C0BFD234D
Content-Type: application/json
Request Body:
{
    "applicationState": "STARTED"
}
```

```
Server Response:
Response Status: 200
Response Body:
{
               "metadata": {
                              "message": "Triggered start of application
process.",
                              "url": "/lifecycle/v1/accounts/test/apps/myapp",
                              "createdAt": 1501825923105,
                              "updatedAt": 1501827428000
               },
               "entity": {
                              "applicationState": "STARTING",
                              "processes": [{
                                             "processId":
"dc1460001710d282b42b7331f1831ec5ad9c1924",
                                             "status": "PENDING",
                                             "lastStatusChange": 0,
                                             "availabilityZone": "",
                                             "computeUnitSize": "LITE"
                              }],
                              "warningMessage": "Triggered start of
application process."
               }
}
```

   The *applicationState* value will change from *STARTING* (or *PENDING*) to *STARTED*.

3. Make sure the application is working properly.

Send a *GET* Application State request to verify whether your application is started. Send this request every 5-10 seconds and check the *applicationState* property in the response. If that property shows the *STARTED* value, then you have successfully started your application:

```
Client Request:
GET: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps/myapp/state
```

```
Server Response:
Response Body:
{
               "metadata": {
                              "domain": "hana.ondemand.com",
                              "aliases": "[\"/DemoApp\",\"example\",\"/\"]",
                              "accessPoints": ["https://
myapptest.int.hana.ondemand.com", "https://myapptest.hana.ondemand.com"],
                              "runtime": {
                                             "id": "neo-java-web",
                                             "state": "recommended",
                                             "expDate": "1541203200000",
                                             "displayName": "Java Web",
                                             "relDate": "1501718400000",
                                             "version": "1.133.3"
                              },
                              "url": "/lifecycle/v1/accounts/test/apps/myapp",
                              "createdAt": 1501825923105,
                              "updatedAt": 1501827428000
               },
               "entity": {
                              "applicationState": "STARTED",
                              "loadBalancerState": "ENABLED",
                              "urls": ["https://
myapptest.int.hana.ondemand.com", "https://myapptest.hana.ondemand.com"],
                              "processes": [{
                                             "processId":
"dc1460001710d282b42b7331f1831ec5ad9c1924",
                                             "status": "STARTED",
                                             "lbStatus": "ENABLED",
                                             "lastStatusChange":
1501827728209,
                                             "runtime": {
                                                            "id": "neo-java-
web",
                                                            "state":
"recommended",
                                                            "expDate":
"1541203200000",
                                                            "displayName":
"Java Web",
                                                            "relDate":
"1501718400000",
                                                            "version":
"1.133.3.2"
                                             },
                                             "availabilityZone": "PRSAG",
                                             "computeUnitSize": "LITE"
                              }]
               }
}
```

4.2.5.1.6 Stop an Application

Procedure

1. Get a CSRF token. If you try to start your application long after its deployment, the token has most probably expired.

   Send a *GET* CSRF Protection request.

   > **i Note**
   >
   > If your session is still actively running, you don't have to request a new CSRF token. In this case, we will use the CSRF token generated during the deployment scenario.

2. Stop the application.

   Send a *PUT* Application State request and set the value of the *applicationState* property to*STOPPED*:

```
Client Request:
PUT: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps/myapp/state
Request Headers:
X-CSRF-Token: 3B95B7A8B0E8E6B923C67E6C0BFD234D
Content-Type: application/json
Request Body:
{
    "applicationState": "STOPPED"
}
```

```
Server Response:
Response Status: 200
Response Body:
{
               "metadata": {
                              "message": "Triggered stop of application
process.",
                              "url": "/lifecycle/v1/accounts/test/apps/myapp",
                              "createdAt": 1501825923105,
                              "updatedAt": 1501827428000
               },
               "entity": {
                              "applicationState": "STOPPING",
                              "processes": [{
                                             "processId":
"dc1460001710d282b42b7331f1831ec5ad9c1924",
                                             "status": "PENDING",
                                             "lastStatusChange": 0,
                                             "availabilityZone": "",
                                             "computeUnitSize": "LITE"
                              }],
                              "warningMessage": "Triggered stop of
application process."
               }
}
```

3. Check if the application is stopped.

   Send a *GET* Application State request. Send this request in a loop every 5-10 seconds and pay close attention to the *applicationState* property in the response. If that property shows the *STOPPED* value, then you have successfully stopped your application.

```
Client Request:
GET: https://api.hana.ondemand.com/lifecycle/v1/accounts/test/apps/myapp/state
```

```
Server Response:
Response Body:
{
    "metadata": {
        "aliases": "[]",
        "runtime": {
            "id": "neo-java-web",
            "state": "recommended",
            "expDate": "1541203200000",
            "displayName": "Java Web",
            "relDate": "1501718400000",
            "version": "1.133.3"
        },
        "url": "/lifecycle/v1/accounts/test/apps/myapp",
        "createdAt": 1502274734263,
        "updatedAt": 1502274835000
    },
    "entity": {
        "applicationState": "STOPPED",
        "processes": []
    }
}
```

Related Information

Java ALM API

4.2.5.2 Deploy Locally with the Console Client

The console client allows you to install a server runtime in a local folder and use it to deploy your application.

Context

Procedure

1. Open the folder `<SDK installation folder>/tools`.
2. Open the command window, enter the following command, and press `ENTER`:

   ```
   neo install-local
   ```

   This installs a server runtime in the default local server directory `<SDK installation folder>/server`. To use an alternative directory, enter the command together with the following optional command argument:

   ```
   --location <local server directory>
   ```

3. To start the local server, enter the following command and press ENTER:

   ```
   neo start-local
   ```

   This starts a local server instance in the default local server directory `<SDK installation folder>/server`. Again, use the following optional command argument to specify another directory:

   ```
   --location <local server directory>
   ```

4. To deploy your application, enter the following command as shown in the example below and press ENTER:

   ```
   neo deploy-local --source hello-world.war
   ```

   This deploys the WAR file on the local server instance. If necessary, specify another directory as in step 3.

5. To check your application is running, open a browser and enter the URL, for example:
   `http://localhost:8080/hello-world`

   > **i Note**
   >
   > The HTTP port is normally 8080. However, the exact port configurations used for your local server, including the HTTP port, are displayed on the console screen when you install and start the local server.

6. To stop the local server instance, enter the following command from the `<SDK installation folder>/tools` folder and press ENTER:

   ```
   neo stop-local
   ```

Related Information

install-local [page 1475]
start-local [page 1560]
deploy-local [page 1428]
stop-local [page 1564]

4.2.5.3 Deploy on the Cloud with the Console Client

Deploying an application publishes it to SAP BTP. During deploy, you can define various specifics of the deployed application using the deploy command optional parameters.

Prerequisites

- You have downloaded and configured SAP BTP console client. For more information, see Set Up the Console Client [page 847]

- Depending on your subaccount type, deploy the application on the respective region host. For more information, see Regions

Context

Procedure

1. In the opened command line console, execute **`neo deploy`** command with the appropriate parameters.

   You can define the parameters of commands directly in the command line as in the example below, or in the properties file. For more information, see Using the Console Client [page 1348].
2. Enter your password if requested.
3. Press ENTER and deployment of your application will start. If deployment fails, check if you have defined the parameters correctly.

> **i** Note
>
> The size of an application deployed on SAP BTP can be up to 1.5 GB. If the application is packaged as a WAR file, the size of the unzipped content is taken into account.

Example

```
neo deploy --host <host> --account <subaccount_name> --application
<application_name> --source samples/deploy_war/example.war --user <email_or_user>
```

Next Steps

To make your deployed application available for requests, you need to start it by executing the **`neo start`** command.

Then, you can manage the application lifecycle (check the status; stop; restart; undeploy) using dedicated console client commands.

Related Information

deploy [page 1422]
start [page 1557]

restart [page 1528]
stop [page 1562]
status [page 1555]
undeploy [page 1574]
Use Delta Deployment [page 893]
Updating Applications [page 1609]
Update Application Properties [page 1593]

4.2.5.3.1 Use Delta Deployment

By using the delta deployment option, you can apply changes in a deployed application faster without uploading the entire set of files to SAP BTP.

Context

The `delta` parameter allows you to deploy only the changes between the provided source and the previously deployed content - new content is added; missing content is deleted; existing content is updated if there are changes. The `delta` parameter is available in two commands – `deploy` and `hot-update`.

> **i Note**
>
> Use it to save time for development purposes only. For updating productive applications, deploy the whole application.

Procedure

To upload only the changed files from the application WARs, use one of the two approaches:

- Deploy the application specifying the `--delta` parameter:

  ```
  neo deploy myapp.properties  --source <file_location> --delta
  ```

  For the changes to take effect, restart the application.
- Upload the delta and apply the changes on an already running application process with the `hot-update` command:

  ```
  neo hot-update myapp.properties  --source <file_location> --strategy
  <update_strategy> --delta
  ```

> **i Note**
>
> With the `source` parameter, provide the whole set of files of your application, not only the changed ones.

Related Information

hot-update [page 1473]
deploy [page 1422]
Deploy on the Cloud with the Console Client [page 891]

4.2.5.4 Deploy on the Cloud with the Cockpit

The cockpit allows you to deploy Java applications as WAR files and supports a number of deployment options for configuring the application.

Context

Procedure

1. Log on to the cockpit and select an account.
2. Choose *Java Applications* in the navigation area.
3. Choose *Deploy Application*.
4. Select the WAR file that you want to deploy, for example, in directory `<SDK_location>/samples/`.
5. Use the application name that the cockpit proposes to you. It is the same name as for the WAR file. Alternatively, enter an application name. Note that application names must start with a letter, can contain lowercase letters and numbers only, and must not exceed 30 characters.

   You can also assign a display name and a description to a Java application.

   > **i Note**
   >
   > When the sum of the characters of the subaccount technical name and the application name exceeds 30, the default application URL is truncated. This means that parts of the subaccount name and the application name will be missing from the application URL:
   >
   > > **Example**
   > >
   > > Subaccount name: mycompanyextensionspoc
   > >
   > > Application name: myappsimplesimplereq
   > >
   > > Application URL: `https://myappsimplemycompanyextensions.us1.hana.ondemand.com`

6. Optionally specify additional parameters to configure the application. If omitted, default values will be assigned.

   For more information about the deploy parameters, see the deploy [page 1422] command documentation.

7. Choose *Deploy* to deploy the WAR file to the cloud platform.
   The *Deploy Application* dialog box remains on the screen while the deployment is in progress. When the deployment is over, a confirmation appears that the application has been successfully deployed. Note that at this stage the application is not yet up and running.
8. In the dialog box, choose one of the following options:
   - *Start*: Start the application to activate its URL and make the application available to your end users.
   - *Close*: Simply close the dialog box if you do not want to start the application immediately.

Results

Your newly deployed application appears in the list of Java applications.

Updating a deployed application

You can update or redeploy the application whenever required. To do this, choose *Update application* to open the same dialog box as in update mode. You can update the application with a new WAR file or change the configuration parameters.

To change the name of a deployed application, deploy a new application under the desired name, and delete the application whose name you want to change.

Related Information

deploy [page 1422]
Define Application Details (Java Apps) [page 1602]

4.2.6 Debugging Applications

After you have created a Web application and tested it locally, you may want to inspect its runtime behavior and state by debugging the application in SAP BTP.

Context

The debugger enables you to detect and diagnose errors in your application. It allows you to control the execution of your program by setting breakpoints, suspending threads, stepping through the code, and examining the contents of the variables. You can debug a servlet or a JSP file on a SAP BTP server without losing the state of your application.

> **i** Note
>
> Currently, it is only possible to debug Web applications in SAP BTP that have exactly one application process (node).

Tasks

- Debug Applications Locally [page 896]
- Debug Applications on the Cloud [page 897]

Related Information

Profiling Applications [page 1618]

4.2.6.1 Debug Applications Locally

In this section, you can learn how to debug a Web application locally.

Prerequisites

- You have developed an application. For more information, see Using Samples [page 850].
- You have installed SAP BTP SDK for the Neo environment. See Install the SAP BTP SDK for Neo Environment [page 845].

Context

Procedure

1. Deploy your application locally using the console client on the Neo environment. For more information, see Deploy Locally with the Console Client [page 890].
2. Before starting the local server, enable remote debugging.
   - For Microsoft Windows:
     Add the following text in the end of your `/<SDK installation folder>/server/bin/setenv.bat`:
     ```
     set CATALINA_OPTS="-agentlib:jdwp=transport=dt_socket,address=<address>,server=y,suspend=n"
     ```
     Where *address* is the debugger port, such as *5005* for up until Java SE 8 and **:5005* for Java SE 9 and later.

- For Linux, Mac OS X, or other Unix based OS:
  Add the following text in the end of your `/<SDK installation folder>/server/bin/setenv.sh`:
  `CATALINA_OPTS="-agentlib:jdwp=transport=dt_socket,address=<address>,server=y,suspend=n"`
  Where *address* is the debugger port, such as *5005* for up until Java SE 8 and **:5005* for Java SE 9 and later.

3. 3. Launch a remote debug session from within your IDE of choice by choosing *localhost* for host and the *address* you have provided previously for port.

Related Information

Debug Applications on the Cloud [page 897]

4.2.6.2 Debug Applications on the Cloud

In this section, you can learn how to debug a Web application on SAP BTP.

Prerequisites

- You have installed Eclipse IDE and the SAP JVM Tools for Eclipse. For more information, see Install the SAP JVM Tools in Eclipse [page 846].
- You have developed a Web application using IDE of your choice. For more information, see Developing Java Applications [page 871].
- You have deployed your Web application. For more information, see Deploying and Updating Applications [page 875].

> **i** Note
>
> Debugging can be enabled if there is only one VM started for the requested account or application.

Context

You can use IDE of your choice for developing the application. You must use Eclipse IDE for debugging the application on the cloud.

Procedure

1. Deploy your Web application in the console client and start it.
2. Go to the Eclipse IDE, open the *Run* menu and choose *Debug Configurations*.
3. Create a new debug configuration of type *SAP HANA Cloud Debugger*.
4. In the *Project* section, choose the Java project that is already imported in the Eclipse IDE.
5. In the *Connection Properties* section, provide the application data.
6. Choose *Debug*.
7. Open the *Debug* perspective in the Eclipse IDE.
8. Set breakpoints in your application.
9. Request your application.

Related Information

Eclipse: Debugging a servlet on a server
Eclipse: Debugging a JSP file on a server

4.2.7 Developing Multitenant Applications in the Neo Environment

In the Neo environment of SAP BTP, you can develop and run multitenant (tenant-aware) applications. These applications run on a shared compute unit that can be used by multiple consumers (tenants). Each consumer accesses the application through a dedicated URL.

You can read about the specifics of each platform service with regards to multitenancy in the respective section below:

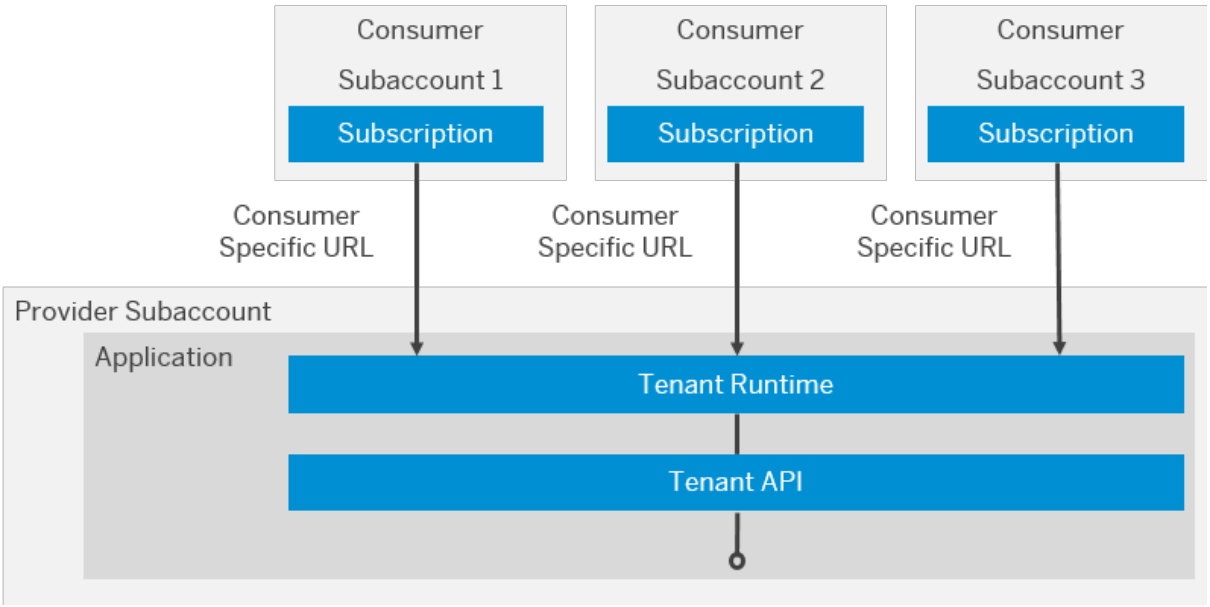
- Connectivity Service [page 899]
- SAP HANA / SAP ASE Service [page 900]
- Keystore Service [page 900]
- Identity and Access Management [page 900]

With tenant-aware applications, you can achieve the following:

- Isolate data
- Save resources by sharing them among tenants
- Perform updates efficiently, that is, in one step

To be able to use a tenant-aware application, a consumer:

- Must have a subaccount for SAP BTP
- Must be subscribed to this application
- Receives a dedicated URL where to access the application



A subscription means that there is a contract between an application provider and a tenant who authorizes the tenant to use the provider's application.

Currently, you can trigger the subscription via the console client. For more information, see Providing Java Multitenant Applications to Tenants for Testing [page 836].

When an application is accessed via a consumer specific URL, the application environment is able to identify the current consumer. The application developer can use the tenant context API to retrieve and distinguish the tenant ID, which is the unique ID of the consumer. When developing tenant-aware applications, data isolation for different consumers is essential. It can be achieved by distinguishing the requests based on the tenant ID. There are also some specifics in the usage of different services when you develop your multitenant application.

For more information, see Tenant Context API [page 902].

General Programming Guidelines

When developing tenant-aware applications, keep in mind the following:

- Shared in-memory data such as Java static fields will be available to all tenants.
- Avoid any possibility that an application user can execute custom code in the application JVM, as this may give them access to other tenants' data.
- Avoid any possibility that an application user can access a file system, as this may give them access to other tenants' data.

Connectivity Service

For more information, see Multitenancy in the Connectivity Service [page 131].

SAP HANA / SAP ASE Service

Multitenant applications on SAP BTP have two approaches available to separate the data of the different consumers:

- **Use a discriminator column in each table storing tenant data**
  With this approach, a single database schema is shared between all application consumers. The tenant ID can be used as a value in the discriminator column. To ensure data separation in the application, each SQL statement must include the tenant ID.
  To apply data separation with a discriminator column, you can use the multitenancy annotations provided by EclipseLink JPA. For more information, see the EclipseLink User Guide: http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Advanced_JPA_Development/Single-Table_Multi-Tenancy .
  The basic approach is as follows:
  - Annotate entities that are to be tenant-aware with the `@Multitenant` annotation.
  - Define the discriminator column using the `@TenantDiscriminatorColumn` annotation. As the length of the tenant ID on SAP BTP differs from the default length for the discriminator column in EclipseLink, it is important to set the correct length of 36 characters.
  - Provide the tenant ID to the entity manager when accessing data for a multitenant entity.
- **Use a separate schema for each tenant**
  With this approach, you create a new schema for each tenant, bind it to the application, and the application uses JNDI to dynamically look up the data source. The multitenant application must then use the appropriate data source when accessing tenant data. For more information, see Apply Dynamic Data Source Lookup [page 955].

Keystore Service

The Keystore Service provides a repository for cryptographic keys and certificates to tenant-aware applications hosted on SAP BTP. Because the tenant defines a specific configuration of an application, you can configure an application to use different keys and certificates for different tenants.

For more information about the Keystore Service, see Keys and Certificates [page 1781].

Identity and Access Management

Access rights for tenant-aware application are usually maintained by the application consumer, not by the application provider. An application provider may predefine roles in the `web.xml` when developing the application. By default, predefined roles are shared with all application consumers, but could also be made visible only to the provider subaccount. Once a consumer is subscribed to this application, shared predefined roles become visible in the cockpit of the application consumer. Then, the application consumer can assign users to these roles to give them access to the provider application. In addition, application consumer subaccounts can add their own custom roles to the subscribed application. Custom roles are visible only within the application consumer subaccount where they are created.

For more information about managing application roles, see Managing Roles [page 1710].

Trust configuration regarding authentication with SAML2.0 protocol is maintained by the application consumer.

For more information about configuring trust, see Application Identity Provider [page 1720].

Related Information

Multitenancy Tutorials [page 905]
Default Trace File [page 1606]
Providing Java Multitenant Applications to Tenants for Testing [page 836]

4.2.7.1 Multitenancy Roles

Context

The multitenancy concept concerns two main user roles:

- `Application Provider` - an organizational unit that uses SAP BTP to build, run and sell applications to customers, that is, the application consumers.
- `Application Consumer` - an organizational unit, typically a customer or a department inside a customer's organization, which uses an SAP BTP application for a certain purpose. Obviously, the application is in fact used by end users, who might be employees of the organization (for instance, in the case of an HR application) or just arbitrary users, internal or external (for instance, in the case of a collaborative supplier application).

Subaccounts and Subaccount Members

To use SAP BTP, both the application provider and the application consumer need to have a subaccount. The subaccount is the central organizational unit in SAP HANA Cloud Plaftorm. It is the central entry point to SAP BTP for both application providers and consumers. It may consist of a set of applications, a set of subaccount members and a subaccount-specific configuration.

Subaccount members are users who must be registered via the SAP ID service. Subaccount members may have different privileges regarding the operations which are possible for a subaccount (for example, subaccount administration, deploy/start/stop applications). Note that the subaccount belongs to an organization and not to an individual. Nevertheless, the interaction with the subaccount is performed by individuals, the members of the subaccount. The subaccount-specific configuration allows application providers and application consumers to adapt their subaccount to their specific environment and needs.

An application resides in exactly one subaccount, the hosting subaccount. It is uniquely identified by the subaccount name and the application name. Applications consume SAP BTP resources, for instance, compute units, structured and unstructured storage and outgoing bandwidth. Costs for consumed resources are billed to the owner of the hosting subaccount, who can be an application provider, an application consumer, or both.

Deployment Application Models

- *Consumer-Managed Application Model*
  In the consumer-managed application model, a copy of the application is deployed into the consumer subaccount. The application consumer buys the application from the application provider via SAP Store. In addition, the application consumer separately buys all required SAP BTP resources (such as compute units, storage or bandwidth) from the platform provider, that is SAP. All platform resources consumed by the application are directly billed to the application consumer. The compute units hosting the application are not shared between multiple application consumers. Instead, each application consumer runs the application on dedicated compute units (VMs). In the consumer-managed application model, the application consumer is responsible to operate the application.
- *Provider-Managed Application Model*
  In the provider-managed application model, the application is deployed into the provider subaccount only. The application provider may decide to start a dedicated compute unit for each consumer or to share one or multiple compute units between them. The application is operated and maintained by the application provider. Platform resources consumed by the application are billed to the application provider. The application provider is responsible for passing the costs on to the consumers. Application providers can define their own price model, which can be independent from the platform resources prices.
  In the provider-managed deployment model, there is no deployment required into the consumer subaccount. Instead, consumers are "subscribed" to the provider application, once they have purchased the application. As a result, a subscription for the purchased application is created in the consumer subaccount. This subscription enables consumers to launch the application, using a consumer-specific URL, and configure it to their needs.
  Typically, the application consumer configures the used identity provider, the user roles for the application, or, if required by the subscribed application, the connection parameters to another system.

Related Information

Providing Java Multitenant Applications to Tenants for Testing [page 836]

4.2.7.2 Tenant Context API

Overview

In a provider-managed application scenario, each application consumer gets its own access URL for the provider application. To be able to use an application with a consumer-specific URL, the consumer must be subscribed to the provider application. When an application is launched via a consumer-specific URL, the tenant runtime is able to identify the current consumer of the application. The tenant runtime provides an API to retrieve the current application consumer. Each application consumer is identified by a unique ID, which is called `tenantId`.

Since the information about the current consumer is extracted from the request URL, the tenant runtime can only provide a tenant ID if the current thread has been started via an HTTP request. In case the current thread wasn't started via an HTTP request (for example, a background process), the tenant context API only returns a tenant if the current application instance has been started for a dedicated consumer. If the current application instance is shared between multiple consumers and the thread wasn't started via an HTTP request, the tenant runtime throws an exception.

> **i Note**
>
> The tenant context API is of interest to application providers only.

Tenant Context API

API	Description
`com.sap.cloud.account`	The tenant context API provides the following methods: • **`getTenant`** - returns the tenant associated with the current thread. • **`execute`** - executes `callable.call()` method on behalf of the specified tenant and returns the result of a called method. • **`getSubscribedTenants`** - returns all tenants subscribed to a calling application, or empty collection if there are no subscribed tenants. To find the tenant context API in your local SDK installation, go to: `<SDK_location>/javadoc/com/sap/cloud/account` For more information, see API Documentation [page 1147].

Tenant Context API via JNDI Lookup

To look up the `TenantContext` API via JNDI, follow the steps:

1. Add the following resource reference of type `com.sap.cloud.account.TenantContext` in your `web.xml` file:

```
<resource-ref>
    <res-ref-name>TenantContext</res-ref-name>
    <res-type>com.sap.cloud.account.TenantContext</res-type>
</resource-ref>
```

2. Then, use the following code to look up:

```
Context ctx = new InitialContext();
TenantContext tenantctx = (TenantContext) ctx.lookup("java:comp/env/
TenantContext");
```

Tenant Context API via Resource Injection

To get an instance of the `TenantContext` API, use resource injection the following way:

```
@Resource
private TenantContext tenantContext;
```

> **i Note**
>
> When you use WebSockets, the `TenantId` and `AccountName` parameters, provided by the `TenantContext` API, are correct only during processing of WebSocket handshake request. This is because what follows after the handshake doesn't conform to the HTTP protocol. In case `TenantId` and `AccountName` are needed during next WebSocket requests, they should be stored into the HTTP session, and, if needed, you can use `TenantContext.execute(...)` to operate on behalf of the relevant tenant.

Account API

The `Account` API provides methods to get subaccount ID, subaccount display name, and attributes.

For more information, see API Documentation [page 1147].

You can access the `Account` API in two ways.

- Via looking up or injecting the `TenantContext` API:

  > **‹≡› Sample Code**
  >
  > ```
  > Context ctx = new InitialContext();
  > TenantContext tenantctx = (TenantContext) ctx.lookup("java:comp/env/
  > TenantContext");
  >
  > Account account = tenantctx.getTenant().getAccount();
  > ```

- Via the `getSubscribedTenants` method:

  > **‹≡› Sample Code**
  >
  > ```
  > Context ctx = new InitialContext();
  > TenantContext tenantctx = (TenantContext) ctx.lookup("java:comp/env/
  > TenantContext");
  >
  > Collection<Tenant> subscribedTenants = tenantctx.getSubscribedTenants();
  >
  > for(Tenant tenant: subscribedTenants ) {
  >     Account subscribedAccount = tenant.getAccount();
  > }
  > ```

Related Information

Developing Multitenant Applications in the Neo Environment [page 898]
Default Trace File [page 1606]
Providing Java Multitenant Applications to Tenants for Testing [page 836]

4.2.7.3 Multitenancy Tutorials

Below are listed tutorials describing end-to-end scenarios with multitenant demo applications:

If you want to	Tutorial
Create a general demo application (servlet)	Create an Exemplary Provider Application (Servlet) [page 905]
Create a general demo application (JSP file)	Create an Exemplary Provider Application (JSP) [page 908]
Create a connectivity demo application	Create a Multitenant Connectivity Application [page 911]
Consume a connectivity demo application	Consume a Multitenant Connectivity Application [page 914]

4.2.7.3.1 Create an Exemplary Provider Application (Servlet)

This tutorial explains how to create a sample application which makes use of the multitenancy concept. That is, you can enable your application to be consumed by users, members of a tenant which is subscribed to this application in a multitenant flavor.

Prerequisites

- You have downloaded and set up your Eclipse IDE, SAP HANA Cloud Tools for Java and SAP BTP SDK for Neo environment. For more information, see Setting Up the Development Environment [page 845].
- You are an application provider. For more information, see Multitenancy Roles [page 901].

Context

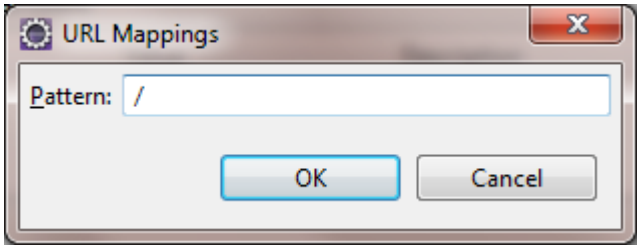
Procedure

1. Create a dynamic Web project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. In the *Project Explorer* view, from the context menu, choose *New* > *Dynamic Web Project*.
3. Enter **`TenantContextApp`** as the *Project name*.
4. In the *Target Runtime* pane, select the runtime you want to use to deploy the application. In this tutorial, we choose **`Java Web`**.
5. In the *Configuration* pane, leave the default configuration and check the option *Generate web.xml deployment descriptor*.
6. Choose *Finish* to finalize the creation of your project.

2. Create a sample servlet

1. From the *TenantContextApp* context menu, choose *New* > *Servlet*.
2. Enter **`tenantcontext.demo`** as the Java package and **`TenantContextServlet`** as the *Class name* and choose *Next*.
3. In the *URL mappings* field, select */TenantContext* and choose *Edit*.
4. In the *Pattern* field, replace the current value with just "**`/`**". In this way, the servlet will be mapped as a welcome page for the application.



5. Choose *Finish* so that the *TenantContext.java* servlet is created and opened in the Java editor.
6. Go to `/TenantContextApp/WebContent/WEB-INF` and open the *web.xml* file.
7. Choose the *Source* tab page.
8. Add the following code block to the `<web-app>` element:

```
<resource-ref>
    <res-ref-name>TenantContext</res-ref-name>
    <res-type>com.sap.cloud.account.TenantContext</res-type>
</resource-ref>
```

9. Replace the entire servlet class with the following sample code:

```
package tenantcontext.demo;
import java.io.IOException;
import java.io.PrintWriter;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.sap.cloud.account.TenantContext;
/**
 * Servlet implementation class TenantContextServlet
 */
public class TenantContextServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public TenantContextServlet() {
        super();
    }
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        try {
            InitialContext ctx = new InitialContext();
            Context envCtx = (Context)ctx.lookup("java:comp/env");
            TenantContext tenantContext = (TenantContext)
envCtx.lookup("TenantContext");
            response.setContentType("text/html");
            PrintWriter writer = response.getWriter();
            writer.println("<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01
Transitional//EN\" \"http://www.w3.org/TR/html4/loose.dtd\">");
            writer.println("<html>");
            writer.println("<head>");
            writer.println("<title>SAP BTP - Tenant Context Demo Application</
title>");
            writer.println("</head>");
            writer.println("<body>");
            writer.println("<h2> Welcome to the SAP BTP Tenant Context demo
application</h2>");
            writer.println("<br></br>");
            String currentTenantId = tenantContext.getTenant().getId();
            writer.println("<p><font size=\"5\"> The application was accessed
on behalf of a tenant with an ID: <b>" + currentTenantId + "</b></font></p>");
            writer.println("</body>");
            writer.println("</html>");
        } catch (Exception e) {
            throw new ServletException(e.getCause());
        }
    }
    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
     */
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        doGet(req, resp);
    }
}
```

10. Save the Java editor. The project compiles without errors.

You have successfully created a Web application containing a sample servlet and connectivity functionality.

3. Deploy the application

To learn how to deploy your application, see Deploy on the Cloud with the Cockpit [page 894].

Result

You have created a sample application that can be requested in a browser. Its output depends on the tenant context.

Next Steps

- To test the access to your multitenant application, go to a browser and request it on behalf of your subaccount. Use the following URL pattern: `https://<application_name><provider_subaccount>.<host>/<application_path>`
- If you want to test the access to your multitenant application on behalf of a consumer subaccount, follow the steps in page: Consume a Multitenant Connectivity Application [page 914]

Related Information

Developing Java Applications [page 871]

Developing Multitenant Applications in the Neo Environment [page 898]

4.2.7.3.2 Create an Exemplary Provider Application (JSP)

This tutorial explains how to create a sample application which makes use of the multitenancy concept. That is, you can enable your application to be consumed by users, members of a tenant which is subscribed to this application in a multitenant flavor.

Prerequisites

- You have downloaded and set up your Eclipse IDE, SAP HANA Cloud Tools for Java and SAP HANA SDK. For more information, see Setting Up the Development Environment [page 845].
- You are an application provider. For more information, see Multitenancy Roles [page 901].

Context

Procedure

1. Create a dynamic Web project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. In the *Project Explorer* view, from the context menu, choose *New* > *Dynamic Web Project*.
3. Enter **`TenantContextApp`** as the *Project name*.
4. In the *Target Runtime* pane, select the runtime you want to use to deploy the application. In this tutorial, we choose **`Java Web`**.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to finalize the creation of your project.

2. Add resource declaration in the Web deployment descriptor

1. Go to `/TenantContextApp/WebContent/WEB-INF` and open the *web.xml* file.
2. Choose the *Source* tab page.
3. Add the following code block to the `<web-app>` element:

```
<resource-ref>
    <res-ref-name>TenantContext</res-ref-name>
    <res-type>com.sap.cloud.account.TenantContext</res-type>
</resource-ref>
```

3. Create a sample JSP

1. Under the *TenantContextApp* project node, choose *New* > *JSP File* in the context menu.
2. Enter **`index.jsp`** as the *File* name and choose *Finish*.
3. Open the *index.jsp* file using the text editor.
4. Replace the entire JSP file content with the following sample code:

```
<%@page
import="javax.naming.InitialContext,javax.naming.Context,com.sap.cloud.account
.TenantContext" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>SAP BTP - Tenant Context Demo Application</title>
</head>
<body>
<h2> Welcome to the SAP BTP Tenant Context demo application</h2>
<br></br>
    <%
        try {
            InitialContext ctx = new InitialContext();
            Context envCtx = (Context) ctx.lookup("java:comp/env");
            TenantContext tenantContext = (TenantContext) envCtx
                    .lookup("TenantContext");
            String currentTenantId = tenantContext.getTenant().getId();
            out.println("<p><font size=\"5\"> The application was accessed on
behalf of a tenant with an ID: <b>"
                    + currentTenantId + "</b></font></p>");
        } catch (Exception e) {
            out.println("error at client");
        }
    %>
</body>
</html>
```

5. Save the Java editor. The project compiles without errors.

4. Deploy the application

To learn how to deploy your application, see .

Result

You have successfully created a Web application containing a JSP file and tenant context functionality.

Next Steps

- To test the access to your multitenant application, go to a browser and request it on behalf of your subaccount. Use the following URL pattern: `https://<application_name><provider_subaccount>.<host>/<application_path>`
- If you want to test the access to your multitenant application on behalf of a consumer subaccount, follow the steps in page: Consume a Multitenant Connectivity Application [page 914]

Related Information

Developing Java Applications [page 871]

Developing Multitenant Applications in the Neo Environment [page 898]

4.2.7.3.3 Create a Multitenant Connectivity Application

Prerequisites

- You have downloaded and set up your Eclipse IDE, SAP BTP Tools for Java and SAP BTP SDK for Neo environment. For more information, see Setting Up the Development Environment [page 845].
- You are an application provider. For more information, see Multitenancy Roles [page 901].

Context

This tutorial explains how you can create a sample application which is based on the multitenancy concept, makes use of the connectivity service, and can be later consumed by other users. That means, you can enable your application to be consumed by users, members of a tenant which is subscribed for this application in a multitenant flavor. The output of the application you are about to create, displays a welcome page showing the URI of the tenant-specific destination configuration. This means that the administrator of the consumer subaccount may have been previously set a tenant-specific configuration for this application. However, in case such configuration has not been set, the application would use the default one, set by the administrator of the provider subaccount.

The application code is the same as for a standard `HelloWorld` consuming the connectivity service as the latter manages the multitenancy with no additional actions required by you. The users of the consumer subaccount, which is subscribed to this application, can access the application using a tenant-specific URL. This would lead the application to use a tenant-specific destination configuration. For more information, see Multitenancy in the Connectivity Service [page 131].

> **i** Note
>
> As a provider, you can set your destination configuration on **application** and **subaccount** level. They are the default destination configurations in case a consumer has not configured tenant-specific destination configuration (on subscription level).

Procedure

1. Create a dynamic Web project

1. Open the *Java EE* perspective of the Eclipse IDE.
2. In the *Project Explorer* view, from the context menu, choose *New* > *Dynamic Web Project*.
3. Enter **`MultitenantConnectivity`** as the *Project name*.
4. In the *Target Runtime* pane, select the runtime you want to use to deploy the application. In this tutorial, we choose **`Java Web`**.
5. In the *Configuration* pane, leave the default configuration.
6. Choose *Finish* to finalize the creation of your project.

2. Add resource declaration in file "Web deployment descriptor"

1. Go to `/MultitenantConnectivity/WebContent/WEB-INF` and open the *web.xml* file.
2. Choose the *Source* tab page.
3. Add the following code block to the `<web-app>` element:

```
<resource-ref>
    <res-ref-name>search_engine_destination</res-ref-name>
    <res-type>com.sap.core.connectivity.api.http.HttpDestination</res-type>
</resource-ref>
```

3. Create a sample JSP file

1. Under the *MultitenantConnectivity* project node, choose *New* > *JSP File* in the context menu.
2. Enter **`index.jsp`** as the *File* name and choose *Finish*.
3. Open the *index.jsp* file using the text editor.
4. Replace the entire JSP file content with the following sample code:

```
<%@page

import="javax.naming.InitialContext,javax.naming.Context,com.sap.core.connecti
vity.api.http.HttpDestination,java.util.Arrays"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>SAP BTP - Multitenant Connectivity Demo Application</title>
</head>
<body>
    <h2>Welcome to SAP BTP - multitenant connectivity demo application</h2>
    <br></br>
    <%
        try {
            Context context = (Context) new InitialContext()
                    .lookup("java:comp/env");
            // In this case you don't need to explicitly use the
TenantContext API
            // because the Connectivity service handles the tenancy by itself.
            // The retrieved HttpDestination object will be tenant-specific.
            String destinationName = "search_engine_destination";
            HttpDestination destination = (HttpDestination) context
                    .lookup(destinationName);
            out.println("<p><font size=\"5\"> Retreived destination with name
<i>"
                    + destination.getName()
                    + "</i> and URI <b>"
                    + destination.getURI() + "</b></font></p>");
        } catch (Exception e) {
            out.println("<b>An exception has been thrown: <i>" +
e.getMessage()
                    + "</i></b>");
            out.println(Arrays.toString(e.getStackTrace()));
        }
    %>
</body>
```

```
</html>
```

5. Save the JSP file.
   The project compiles without errors.

You have successfully created a Web application containing a sample JSP file and consuming the connectivity service via looking up a destination configuration.

4. Deploy the application

To learn how to deploy your application, see .

5. Define the destination configurations

You, as application provider, can configure a default destination, which is then used at runtime when the application is requested in the context of the provider subaccount. In this case, the URL used to access the application is not tenant-specific.

Example:

```
Name=search_engine_destination
URL=https://www.google.com
Type=HTTP
ProxyType=Internet
Authentication=NoAuthentication
TrustAll=true
```

For more information on how to define a destination for provider subaccount, see:

Configure Destinations from the Console Client [page 56].

Configure Destinations from the Cockpit [page 77]

Result

You have created a sample application which can be requested in a browser. Its output depends on the tenant name.

Next Steps

- To test the access to your multitenant application, go to a browser and request it on behalf of your subaccount. Use the following URL pattern: `https://<application_name><provider_subaccount>.<host>/<application_path>`
- If you want to test the access to your multitenant application on behalf of a consumer subaccount, follow the steps in page: Consume a Multitenant Connectivity Application [page 914]

Related Information

Upload Destinations [page 58]
Developing Multitenant Applications in the Neo Environment [page 898]

4.2.7.3.4 Consume a Multitenant Connectivity Application

Prerequisites

- Your SAP BTP user is a member of a subaccount.
- You are subscribed to a provider application, hosted by a provider subaccount. For more information, see Multitenancy Roles [page 901] and Providing Java Multitenant Applications to Tenants for Testing [page 836].

> **i Note**
>
> This tutorial assumes that your subaccount is subscribed to the following exemplary application (deployed in a provider subaccount): Create a Multitenant Connectivity Application [page 911]

Context

This tutorial explains how you can consume a sample connectivity application based on the multitenancy concept. That is, you are a member of a subaccount which is subscribed for applications provided by other subaccounts. The output of the application you are about to consume, displays a welcome page showing the URI of the tenant-specific destination configuration. This means that the administrator of your consumer subaccount may have been previously set a tenant-specific configuration for this application. However, in case such configuration has not been set, the application would use a default one, set by the administrator of the provider subaccount.

Users of a consumer subaccount, which is subscribed to an application, can access the application using a tenant-specific URL. This would lead the application to use a tenant-specific destination configuration. For more information, see Multitenancy in the Connectivity Service [page 131].

> **i Note**
>
> As a consumer, you can set a tenant-specific destination configuration on **subscription** level.

Procedure

1. Define destination configuration

You can consume a provider application if your subaccount is subscribed to it. In this case, administrators of your consumer subaccount can configure a tenant-specific destination configuration, which can later be used by the provider application.

To illustrate the tenant-specific consumption, the URL used in this example is diferent from the one in the exemplary provider application tutorial.

Example:

```
Name=search_engine_destination
URL=http://www.yahoo.com
Type=HTTP
ProxyType=Internet
Authentication=NoAuthentication
TrustAll=true
```

> → Tip
>
> The destination name depends on the provider application.
>
> For more information on how to configure a destination for provider subaccount, see:
>
> Configure Destinations from the Console Client [page 56].
>
> Configure Destinations from the Cockpit [page 77]

2. Access the application

Go to a browser and request the application on behalf of your subaccount. Use the following URL pattern:

```
https://<application_name><provider_subaccount>-<consumer_subaccount>.<host>/
<application_path>
```

Result

The application is requested in a browser. Its output is relevant to your tenant-specific destination configuration.

Related Information

Create a Multitenant Connectivity Application [page 911]
Configure Destinations from the Console Client [page 56]

Configure Destinations from the Eclipse IDE [page 64]
Configure Destinations from the Cockpit [page 77]

4.2.8 Programming with JPA and JDBC

An overview of the options you have when programming with the SAP HANA and SAP ASE databases.

Accessing Databases Remotely [page 916]
: Access remote database instances through a database tunnel, which provides a secure connection from your local machine and bypasses any firewalls.

Programming with JPA [page 917]
: Program with JPA in the Neo environment, whose container-managed persistence and application-managed persistence differ in terms of the management and life cycle of the entity manager.

Programming with Plain JDBC [page 964]
: Program with JDBC in the Neo environment in cases in which its low-level control is more appropriate than JPA.

Testing on the Local Runtime [page 978]
: Test an application in the Neo environment that uses the default data source and runs locally on Apache Derby on the local runtime.

Investigating Performance Issues Using the SQL Trace [page 979]
: Identify inefficient SQL statements in your applications in the Neo environment and investigate performance issues.

Related Information

Console Client Commands

4.2.8.1 Accessing Databases Remotely

Access remote database instances through a database tunnel, which provides a secure connection from your local machine and bypasses any firewalls.

To learn how to access your database remotely, see

- Accessing Databases Remotely for SAP HANA
- Accessing Databases Remotely for SAP ASE

4.2.8.2 Programming with JPA

Program with JPA in the Neo environment, whose container-managed persistence and application-managed persistence differ in terms of the management and life cycle of the entity manager.

The main features of each scenario are shown in the table below. We recommend that you use container-managed persistence (Java EE 6 Web Profile runtime), which is the model most commonly used by Web applications.

JPA Scenario	SDK for Java Web	SDK for Java EE 6 Web Profile
Container-managed persistence	Not supported	JTA transactions
		Entity manager injection using the `@PersistenceContext` annotation
Application-managed persistence	Resource-local transactions	
	Not supported	Injection of the entity manager factory using the `@PersistenceUnit` annotation
	Manual creation of the entity manager factory using `javax.persistence.Persistence. createEntityManagerFactory`	

EclipseLink

Download the latest version of EclipseLink. EclipseLink versions 2.5 and later contain the SAP HANA database platform.

The following JAR files are required:

JPA Scenario	SDK	EclipseLink JARs
Container-managed persistence	Java EE 6 Web Profile	`eclipselink\jlib\eclipselink.jar` Required only for EclipseLink versions 2.5 and later.
Application-managed persistence	Java Web	• `eclipselink\jlib\eclipselink.jar` • `eclipselink\jlib\jpa \javax.persistence_2.*.jar`
	Java EE 6 Web Profile	`eclipselink\jlib\eclipselink.jar`

The EclipseLink download is available from http://www.eclipse.org/eclipselink/downloads . Click *EclipseLink 2.5.x Installer Zip*, then download the archive and extract it to your local file system. For the Java Web scenario, we recommend that you copy the two files (see above) to a separate directory in your local file system.

For details about importing the files into your Web application project and specifying the JPA implementation library EclipseLink, see the tutorial Tutorial: Adding Application-Managed Persistence with JPA (SDK for Java Web) [page 931].

Related Information

Special Settings for EclipseLink Versions Earlier than 2.5 [page 945]
Persistence Units [page 946]
Using Container-Managed Persistence [page 947]
Using Application-Managed Persistence [page 950]
Entity Classes [page 957]

4.2.8.2.1 Tutorial: Adding Container-Managed Persistence with JPA (SDK for Java EE 7 Web Profile)

Use JPA together with EJB to apply container-managed persistence in a simple Java EE web application that manages a list of persons.

Prerequisites

- Download and set up your Eclipse IDE, SAP BTP Tools for Java, and the **SDK for Java EE 7 Web Profile**. For more information, see Setting Up the Development Environment [page 845].
- Set up your runtime environment in the Eclipse IDE.
- Develop or import a Java Web application in Eclipse IDE. For more information, see Developing Java Applications [page 871].
  The application is also available as a sample in the SAP BTP SDK for Neo environment for Java EE 7 Web Profile:
  - Sample name: `persistence-with-ejb`
  - Location: `<sdk>/samples` folder

  More information, see Using Samples [page 850].

Context

Perform the following steps:

1. Create a Dynamic Web Project and Servlet [page 919]
2. Create the JPA Persistence Entity [page 922]
3. Configure the persistence.xml File of the Person Entity [page 923]
4. Create an EJB Session Bean [page 923]
5. Prepare the Web Application Project for JPA [page 924]
6. Extend the Servlet to Use Persistence [page 924]
7. Test the Web Application on the Local Server [page 927]
8. Deploy Applications Using Persistence on the Cloud from Eclipse [page 928]

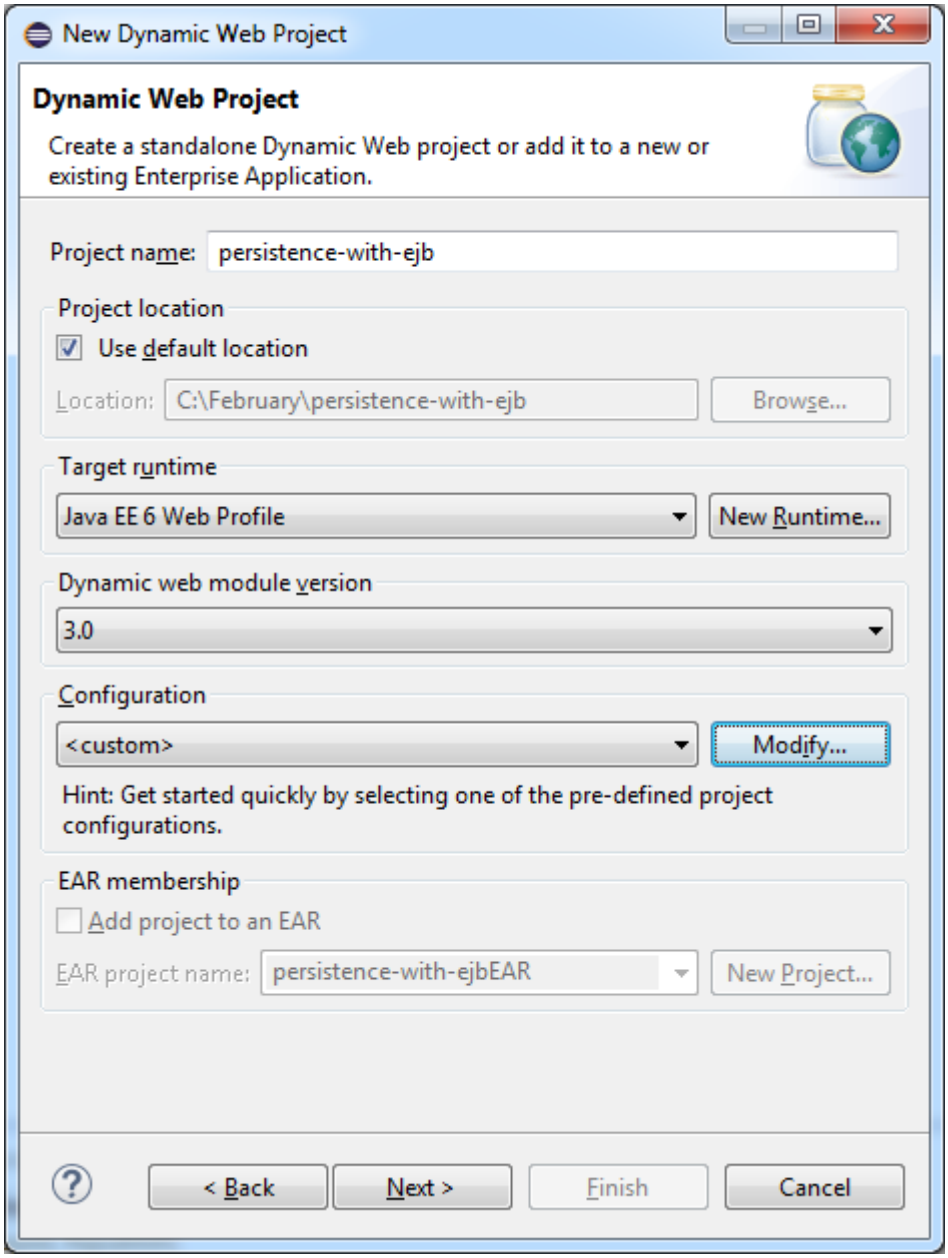
9. Configure Applications Using the Cockpit [page 930]
10. Start Applications Using Eclipse [page 930]

4.2.8.2.1.1 Create a Dynamic Web Project and Servlet

Create a dynamic web project using the JPA project facet and add a servlet.

Procedure

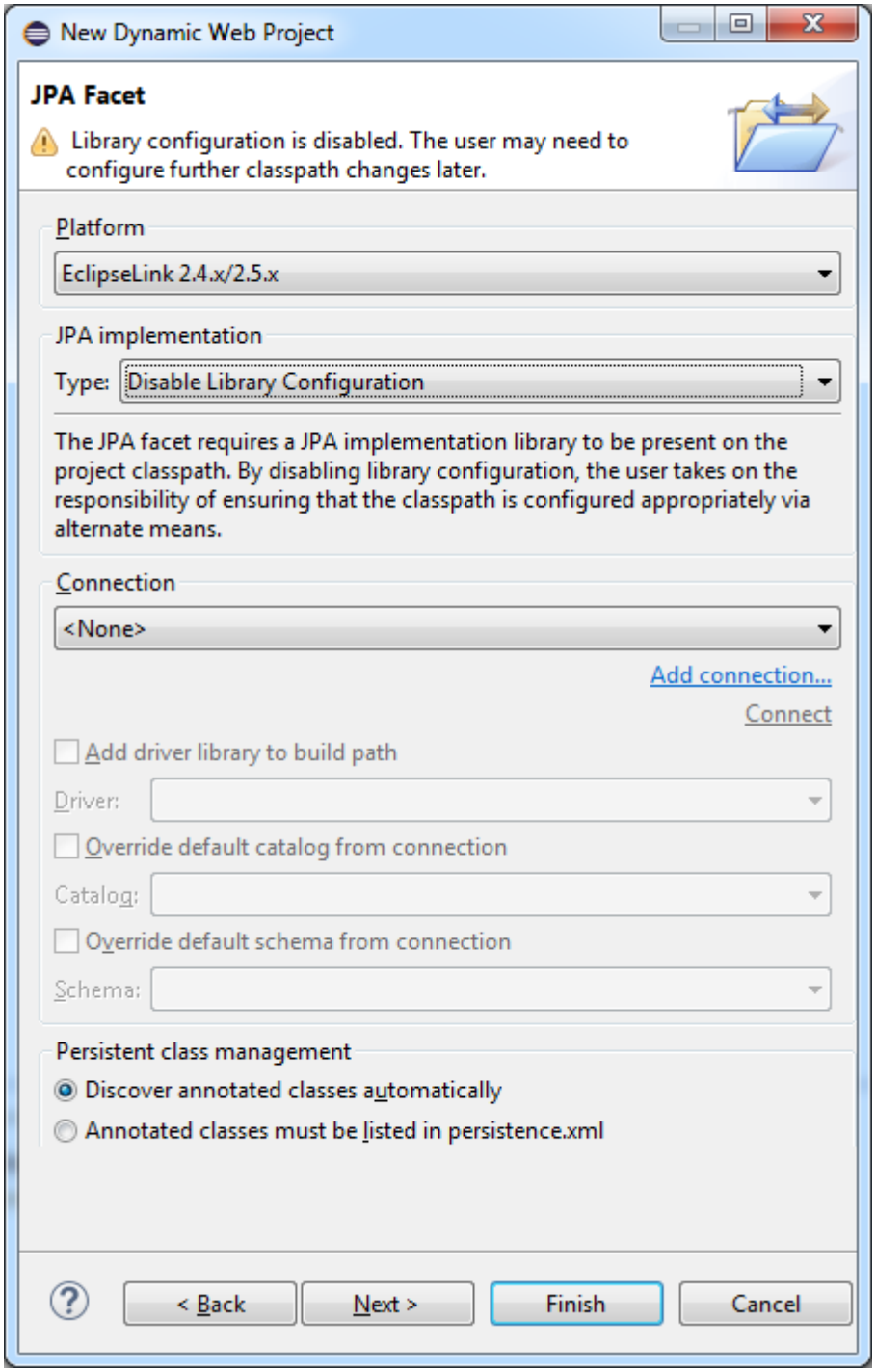
1. From the Eclipse main menu, choose *File* > *New* > *Dynamic Web Project*.
2. Enter the *Project name* `persistence-with-ejb`.
3. In the *Target Runtime* pane, select *Java EE 7 Web Profile* as the runtime you want to use to deploy the application.
4. In the *Dynamic web module version* section, select `3.0`.
5. In the *Configuration* section, choose *Modify* and select *JPA* in the *Project Facets* screen.
6. Choose *OK* and return to the *Dynamic Web Project* screen.
7. Choose *Next*.



8. On the *Java* screen, leave the default settings and choose *Next*.

   You see the *JPA Facet*.

9. In the *Platform* section, select **`EclipseLink 2.4.x/2.5.x`**.
10. In the *JPA implementation* section, select **`Disable Library Configuration`**.
11. In the *Persistent class management* section, make sure that *Discover annotated classes automatically* is selected.



12. Choose *Next*.
13. In the *Web Module* configuration settings, select *Generate web.xml deployment descriptor* and choose *Finish*.
14. To add a servlet to your project, choose *File* > *New* > *Servlet* from the Eclipse main menu.
15. Enter the Java package `com.sap.cloud.sample.persistence` and the class name `PersistenceEJBServlet`.
16. To generate the servlet, choose *Finish*.

4.2.8.2.1.2 Create the JPA Persistence Entity

Procedure

1. In the *Project Explorer* view, select the `persistence-with-ejb/Java Resources/src/com.sap.cloud.sample.persistence` node.
2. From the Eclipse main menu, choose *File* > *New* > *Other* > *Class* and choose *Next*.
3. Make sure that the Java package is **`com.sap.cloud.sample.persistence`**.
4. Enter the class name **`Person`** and choose *Finish*. Replace the entire class with the following content:

```
package com.sap.cloud.sample.persistence;
import javax.persistence.Basic;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
/**
 * Class holding information on a person.
 */
@Entity
@Table(name = "T_PERSON")
@NamedQuery(name = "AllPersons", query = "select p from Person p")
public class Person {
    @Id
    @GeneratedValue
    private Long id;
    @Basic
    private String firstName;
    @Basic
    private String lastName;
    public long getId() {
        return id;
    }
    public void setId(long newId) {
        this.id = newId;
    }
    public String getFirstName() {
        return this.firstName;
    }
    public void setFirstName(String newFirstName) {
        this.firstName = newFirstName;
    }
    public String getLastName() {
        return this.lastName;
    }
    public void setLastName(String newLastName) {
        this.lastName = newLastName;
    }
}
```

5. Save the class.

4.2.8.2.1.3 Configure the persistence.xml File of the Person Entity

In the `persistence-with-ejb/Java Resources/src/META-INF` folder, define additional settings in the `persistence.xml` file.

Procedure

1. Select `persistence.xml`, and from the context menu choose *Open With* > *Persistence XML Editor*.
2. On the *General* tab, make sure that **`org.eclipse.persistence.jpa.PersistenceProvider`** is entered in the *Persistence provider* field.
3. On the *Options* tab, make sure that the DDL generation type **`Create Tables`** is selected.
4. On the *Connection* tab, select the transaction type **`JTA`**.
5. Save the file.

4.2.8.2.1.4 Create an EJB Session Bean

Create an EJB session bean to handle the database operations.

Procedure

1. In the *Project Explorer* view, select the `persistence-with-ejb/Java Resources/src/com.sap.cloud.sample.persistence` node.
2. From the Eclipse main menu, choose *File* > *New* > *Other* > *EJB* > *Session Bean (EJB 3.x)* and choose *Next*.
3. Make sure that the Java package is **`com.sap.cloud.sample.persistence`**.
4. Enter the class name **`PersonBean`**, leave the default setting **`Stateless`**, and choose *Finish*.
5. Leave the default setting **`Stateless`** and choose *Finish*.
6. Replace the entire class with the following content:

```
package com.sap.cloud.sample.persistence;
import java.util.List;
import javax.ejb.LocalBean;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
/**
 * Session Bean implementation class PersonBean
 */
@Stateless
@LocalBean
public class PersonBean {
```

```
    @PersistenceContext
    private EntityManager em;
    public List<Person> getAllPersons() {
        return em.createNamedQuery("AllPersons").getResultList();
    }
    public void addPerson(Person person) {
        em.persist(person);
        em.flush();
    }
}
```

7. Save the class.

4.2.8.2.1.5 Prepare the Web Application Project for JPA

Add the XSS Protection Library to the web application project.

Procedure

1. In the *Project Explorer* view, select the `persistence-with-ejb/WebContent/WEB-INF/lib` node.
2. From the context menu, choose *Import* > *General* > *File System* and choose *Next*.
3. Browse to the local directory where you downloaded and unpacked the SDK for Java EE 7 Web Profile, select the `repository/plugins` directory, and choose *OK*.
4. Select *com.sap.security.core.server.csi_1.x.y.jar* and choose *Finish*.

4.2.8.2.1.6 Extend the Servlet to Use Persistence

Extend the servlet to use the **`Person`** entity and EJB session bean.

Context

The servlet adds **`Person`** entity objects to the database, retrieves their details, and shows them on the screen.

Procedure

1. In the *Project Explorer* view, expand the `persistence-with-ejb/Java Resources/src/com.sap.cloud.sample.persistence` node.
2. Select `PersistenceEJBServlet.java`, and from the context menu choose *Open With* > *Java Editor*.

3. Replace the entire servlet class with the following content:

```
package com.sap.cloud.sample.persistence;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;
import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.sap.security.core.server.csi.IXSSEncoder;
import com.sap.security.core.server.csi.XSSEncoder;
/**
 * Servlet implementation class PersistenceEJBServlet
 */
@WebServlet("/")
public class PersistenceEJBServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final Logger LOGGER = LoggerFactory
            .getLogger(PersistenceEJBServlet.class);
    @EJB
    PersonBean personBean;
    /** {@inheritDoc} */
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        response.getWriter().println("<p>Persistence with JPA!</p>");
        try {
            appendPersonTable(response);
            appendAddForm(response);
        } catch (Exception e) {
            response.getWriter().println(
                    "Persistence operation failed with reason: "
                            + e.getMessage());
            LOGGER.error("Persistence operation failed", e);
        }
    }
    /** {@inheritDoc} */
    @Override
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        try {
            doAdd(request);
            doGet(request, response);
        } catch (Exception e) {
            response.getWriter().println(
                    "Persistence operation failed with reason: "
                            + e.getMessage());
            LOGGER.error("Persistence operation failed", e);
        }
    }
    private void appendPersonTable(HttpServletResponse response)
            throws SQLException, IOException {
        // Append table that lists all persons
        List<Person> resultList = personBean.getAllPersons();
        response.getWriter().println(
                "<p><table border=\"1\"><tr><th colspan=\"3\">"
                        + (resultList.isEmpty() ? "" : resultList.size()
                                + " ")
                                + "Entries in the Database</th></tr>");
        if (resultList.isEmpty()) {
```

```
            response.getWriter().println(
                    "<tr><td colspan=\"3\">Database is empty</td></tr>");
        } else {
            response.getWriter()
            .println(
                    "<tr><th>First name</th><th>Last name</th><th>Id</th></
tr>");
        }
        IXSSEncoder xssEncoder = XSSEncoder.getInstance();
        for (Person p : resultList) {
            response.getWriter().println(
                    "<tr><td>" + xssEncoder.encodeHTML(p.getFirstName())
                    + "</td><td>"
                    + xssEncoder.encodeHTML(p.getLastName())
                    + "</td><td>" + p.getId() + "</td></tr>");
        }
        response.getWriter().println("</table></p>");
    }
    private void appendAddForm(HttpServletResponse response) throws
IOException {
        // Append form through which new persons can be added
        response.getWriter()
                .println(
                        "<p><form action=\"\" method=\"post\">"
                                + "First name:<input type=\"text\" name=
\"FirstName\">"
                                + " Last name:<input type=\"text\" name=
\"LastName\">"
                                + " <input type=\"submit\" value=\"Add
Person\">"
                                + "</form></p>");
    }
    private void doAdd(HttpServletRequest request) throws ServletException,
            IOException, SQLException {
        // Extract name of person to be added from request
        String firstName = request.getParameter("FirstName");
        String lastName = request.getParameter("LastName");
        if (firstName != null && lastName != null
                && !firstName.trim().isEmpty()
                && !lastName.trim().isEmpty()) {
            Person person = new Person();
            person.setFirstName(firstName);
            person.setLastName(lastName);
            personBean.addPerson(person);
        }
    }
}
```

4. Save the servlet.

Results

The project should compile without any errors.

4.2.8.2.1.7 Test the Web Application on the Local Server

Test your application by deploying it locally.

Procedure

1. To test your web application on the local server, follow the steps for deploying a web application locally as described in .

   You should see the following output:

2. Enter a first name (for example, `John`) and a last name (for example, `Smith`) and choose *Add Person*.

   `John Smith` is added to the database as shown below:

> **i Note**
>
> If you add more names to the database, they are also listed in the table. This confirms that you have successfully enabled persistence using the `Person` entity.

4.2.8.2.1.8 Deploy Applications Using Persistence on the Cloud from Eclipse

To test your web application in the cloud, define a server in Eclipse.

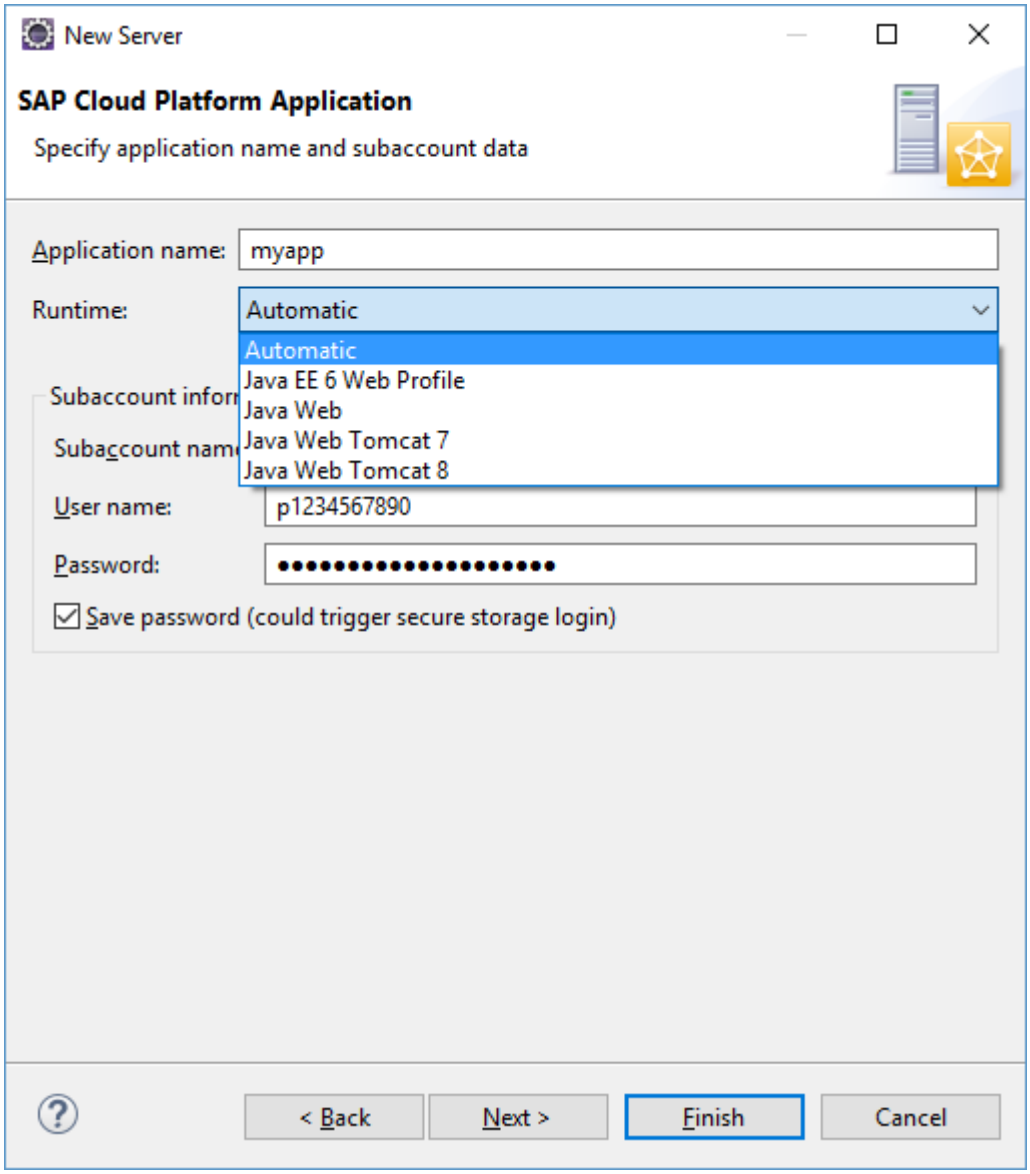
Procedure

1. In the Eclipse IDE, switch to the *Servers* view.
2. Open the context menu define a server with the following settings:
3. Select the server type *SAP* > *SAP Cloud Platform*.
4. Use the host depending on your global account type, then choose *Next*. For more information, see Regions.
5. Specify a unique and identifiable application name.

   Only lowercase Latin letters and digits are allowed.
6. Select a runtime.

   If you leave the default `Automatic` option, the server loads the target runtime of your application.
7. Enter your subaccount name, e-mail or user name, and password, then choose *Next*.
   - If you have previously entered a subaccount and user name for your host, you can select these names from lists.
   - Previously entered hosts also appear in a dropdown list.
   - Select *Save password* to remember and store the password for a given user.

> ⚠ Caution
>
> Do not select your application on the *Add and Remove* screen. Adding an application automatically starts it with the effect that it will fail because no data source binding exists. You will add an application in a later step.



8. Choose *Finish*.
9. In the *Servers* view, open the context menu for the server you just created and choose *Show In* > *Cockpit*.

   The cockpit opens inside Eclipse.

4.2.8.2.1.9 Configure Applications Using the Cockpit

Use the cockpit to create a default binding for your application.

Procedure

1. In the cockpit, select a subaccount, then choose *SAP HANA / SAP ASE* > *Databases & Schemas* in the navigation area.
2. Select the database you want to create a binding for.
3. Choose *Data Source Bindings*. For more information, see .
4. Define a binding (<DEFAULT>) for the application and select a database ID. Choose *Save*.

   You can use an existing database or create a new one.

4.2.8.2.1.10 Start Applications Using Eclipse

Add the application to the new server and start it to deploy the application on the cloud.

Context

> **i Note**
>
> You cannot deploy multiple applications on the same application process. Deployment of a second application on the same application process overwrites any previous deployments. To deploy several applications, deploy each of them on a separate application process.

Procedure

1. On the *Servers* view in Eclipse, open the context menu for the server choose *Add and Remove...* > *<application name>*.
2. To add the application to the server, add the application to the panel on the right side.
3. Choose *Finish*.
4. Start the server.

   This deploys the application and starts it on the SAP BTP.
5. Access the application by clicking the application URL on the application overview page in the cockpit.

   You should see the same output as when the application was tested on the local server.

4.2.8.2.2 Tutorial: Adding Application-Managed Persistence with JPA (SDK for Java Web)

Use JPA to apply application-managed persistence in a simple Java EE web application that manages a list of persons.

Prerequisites

- Download and set up your Eclipse IDE, SAP BTP Tools for Java, and the SDK for Java Web
  For more information, see Setting Up the Development Environment [page 845].
- Downloaded the JPA Provider, EclipseLink:
  1. Download the latest 2.5.x version of EclipseLink from: http://www.eclipse.org/eclipselink/downloads . Select the *EclipseLink 2.5.x Installer Zip* (intended for use in Java EE environments).

     > → Recommendation
     >
     > We recommend that you use EclipseLink version 2.7.8 as used in the `persistence-with-jpa` sample. The tutorial uses EclipseLink version 2.5 because of the following Eclipse issue .
  2. Extract the archive.
  3. Copy the following two files to a separate directory in your local file system:
     - `eclipselink.jar` from the directory `eclipselink/jlib`
     - `javax.persistence_2.*.jar` from the directory `eclipselink/jlib/jpa`

     You will need to add these files to your web application in a later step.
- Set up your runtime environment in the Eclipse IDE.
- Develop or import a Java Web application in Eclipse IDE. For more information, see Developing Java Applications [page 871].
  The application is also available as a sample in the SAP BTP SDK for Neo environment for Java Web:
  - Sample name: `persistence-with-jpa`
  - Location: `<sdk>/samples` folder

  For more information, see Using Samples [page 850].

Context

Perform the following steps:

1. Create a Dynamic Web Project and Servlet with JPA [page 932]
2. Create the JPA Persistence Entity [page 935]
3. Maintain Metadata for the Person Entity [page 936]
4. Prepare the Web Application Project for JPA [page 937]
5. Extend the Servlet to Use Persistence [page 938]
6. Test the Web Application on the Local Server [page 941]
7. Deploy Applications Using Persistence on the Cloud from Eclipse [page 942]

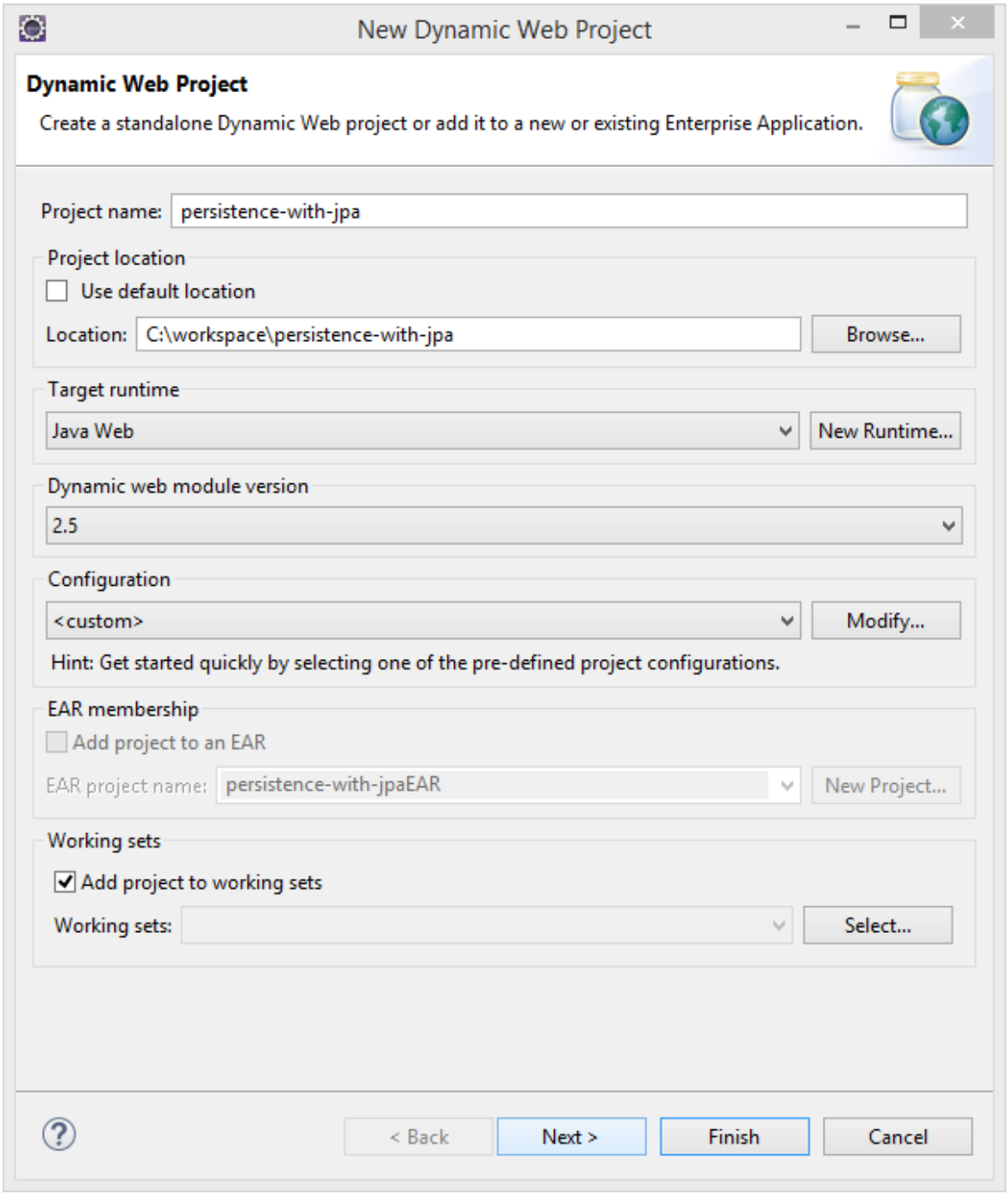
8. Configure Applications Using the Cockpit [page 944]
9. Start Applications Using Eclipse [page 944]

4.2.8.2.2.1 Create a Dynamic Web Project and Servlet with JPA

Create a dynamic web project using the JPA project facet and a servlet.

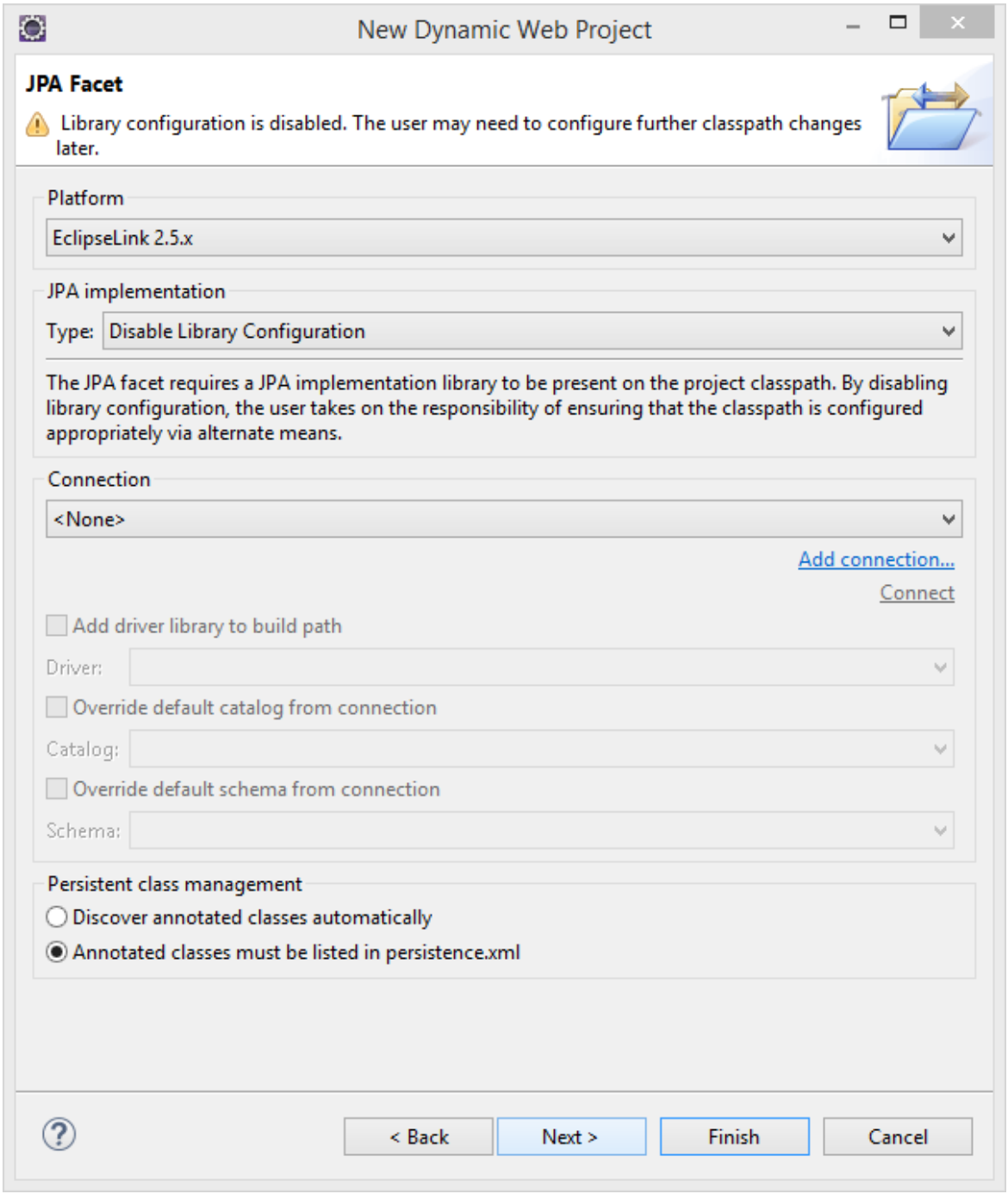
Procedure

1. From the Eclipse main menu, choose *File* > *New* > *Dynamic Web Project*.
2. Enter the *Project name* `persistence-with-jpa`.
3. In the *Target Runtime* pane, select *Java Web* as the runtime to deploy the application.
4. In the *Dynamic web module version* section, select `2.5`.
5. In the *Configuration* section, choose *Modify*, then select the *JPA* in the *Project Facets* screen.
6. Choose *OK* and return to the *Dynamic Web Project* screen.



7. Choose *Next*.
8. On the *Java* screen, leave the default settings and choose *Next*.

   The *JPA Facet* opens.
9. In the *Platform* section, select **`EclipseLink 2.5.x`**.
10. In the *JPA implementation* section, select **`Disable Library Configuration`**.
11. In the *Persistent class management* section, select *Annotated classes must be listed in persistence.xml*.



12. Choose *Next*.
13. In the *Web Module* configuration settings, select *Generate web.xml deployment descriptor* and choose *Finish*.
14. To add a servlet to the project you have just created, choose *File* > *New* > *Servlet* from the Eclipse main menu.
15. Enter the Java package `com.sap.cloud.sample.persistence` and the class name `PersistenceWithJPAServlet`.
16. To generate the servlet, choose *Finish*.

4.2.8.2.2.2 Create the JPA Persistence Entity

Create a JPA persistence entity by replacing the existing class.

Context

Create a JPA persistence entity class named **`Person`**. Add an auto-incremented ID to the database table as the primary key and person attributes. You must also define a query method that retrieves a **`Person`** object from the database table. Each person stored in the database is represented by a **`Person`** entity object.

Procedure

1. In the *Project Explorer* view, select the `persistence-with-ejb/Java Resources/src/com.sap.cloud.sample.persistence` node.
2. From the Eclipse main menu, choose *File* > *New* > *Other* > *Class* and choose *Next*.
3. Make sure that the Java package is **`com.sap.cloud.sample.persistence`**.
4. Enter the class name **`Person`** and choose *Finish*.
5. In the editor, replace the entire class with the following content:

```
package com.sap.cloud.sample.persistence;
import javax.persistence.Basic;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
/**
 * Class holding information on a person.
 */
@Entity
@Table(name = "T_PERSON")
@NamedQuery(name = "AllPersons", query = "select p from Person p")
public class Person {
    @Id
    @GeneratedValue
    private Long id;
    @Basic
    private String firstName;
    @Basic
    private String lastName;
    public long getId() {
        return id;
    }
    public void setId(long newId) {
        this.id = newId;
    }
    public String getFirstName() {
        return this.firstName;
    }
    public void setFirstName(String newFirstName) {
        this.firstName = newFirstName;
    }
```

```
    public String getLastName() {
        return this.lastName;
    }
    public void setLastName(String newLastName) {
        this.lastName = newLastName;
    }
}
```

6. Save the class.

4.2.8.2.2.3 Maintain Metadata for the Person Entity

To maintain metadata for your entity class, define additional settings in the `persistence.xml` file.

Context

In the `persistence-with-jpa/Java Resources/src/META-INF` folder, define additional settings in the `persistence.xml` file.

Procedure

1. Select `persistence.xml` and from the context menu choose *Open With* > *Persistence XML Editor*.
2. Choose the *General* tab.
3. Make sure that **`org.eclipse.persistence.jpa.PersistenceProvider`** is entered in the *Persistence provider* field.
4. In the *Managed Class* section, choose *Add...*, enter *Person*, then choose *Ok*.
5. On the *Connection* tab, make sure that the transaction type **`Resource Local`** is selected.
6. On the *Schema Generation* tab, make sure the DDL generation type **`Create Tables`** in the *EclipseLink Schema Generation* section is selected.
7. Save the file.

4.2.8.2.2.4 Prepare the Web Application Project for JPA

Prepare the web application project by adding EclipseLink executables and the XSS Protection Library, adapting the Java build path order, and adding the resource reference description to the `web.xml` file.

Procedure

1. Add EclipseLink executables to the web application project:
   a. In the *Project Explorer* view, select the `persistence-with-jpa/WebContent/WEB-INF/lib` node.
   b. From the context menu, choose *Import* > *General* > *File System* and choose *Next*.
   c. Browse to the local directory to which you copied the downloaded EclipseLink JAR files (see the Prerequisites [page 931] section), and choose *OK*.
   d. Choose the *Select All* button to select the EclipseLink JAR files, *eclipselink.jar* and *javax.persistence_2.*.jar*. Choose *Finish*.
2. Add the XSS Protection Library to the web application project:
   a. In the *Project Explorer* view, select the `persistence-with-jpa/WebContent/WEB-INF/lib` node.
   b. From the context menu, choose *Import* > *General* > *File System* and choose *Next*.
   c. Browse to the local directory where you downloaded and unpacked the SAP BTP SDK for Java Web (neo-java-web-sdk-1.<version>), select the `repository/plugins` directory, and choose *OK* .
   d. Select *com.sap.security.core.server.csi_1.x.y.jar* and choose *Finish*.
3. Adapt the Java build path order:
   a. In the *Project Explorer* view, select the `persistence-with-jpa` node, and from the context menu choose *Properties*.
   b. Select *Java Build Path* and switch to the *Order and Export* tab.
   c. Select *Web App Libraries* and move it to above `Java Web`.
   d. Choose *OK*.
4. Add the resource reference description to `web.xml`:
   a. In the *Project Explorer* view, expand the `persistence-with-jpa/WebContent/WEB-INF` node.
   b. Select `web.xml` and from the context menu choose *Open With* > *Text Editor*.
   c. Insert the following content after the `<servlet-mapping>` elements:

```
<resource-ref>
    <res-ref-name>jdbc/DefaultDB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
</resource-ref>
```

   d. Save the file.
5. **Optional:** (Optional) Modify the servlet deployment descriptor information:
   a. In the `web.xml` file, replace the URL pattern "/PersistenceWithJPAServlet" that was generated for the servlet with "/" as shown below:

```
<servlet-mapping>
    <servlet-name>PersistenceWithJPAServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

b. Save the file.

> **i Note**
>
> An application's URL path contains the context root followed by the optional URL pattern ("/<URL pattern>"). The servlet URL pattern that is automatically generated by Eclipse uses the servlet's class name as part of the pattern. Since the cockpit shows only the context root, this means that you cannot directly open the application in the cockpit without adding the servlet name. To call the application by only the context root, use "/" as the URL mapping, then you will no longer have to correct the URL in the browser.

4.2.8.2.2.5 Extend the Servlet to Use Persistence

Extend the servlet to use the **`Person`** entity.

Context

The servlet adds **`Person`** entity objects to the database, retrieves their details, and displays them on the screen.

Procedure

1. In the *Project Explorer* view, expand the `persistence-with-jpa/Java Resources/src/com.sap.cloud.sample.persistence` node.
2. Select `PersistenceWithJPAServlet.java` and from the context menu choose *Open With* > *Java Editor*.
3. In the opened editor, replace the entire servlet class with the following content:

```
package com.sap.cloud.sample.persistence;
import java.io.IOException;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;
import org.eclipse.persistence.config.PersistenceUnitProperties;
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.sap.security.core.server.csi.IXSSEncoder;
import com.sap.security.core.server.csi.XSSEncoder;
/**
 * Servlet implementing a simple JPA based persistence sample application for
SAP BTP.
 */
public class PersistenceWithJPAServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final Logger LOGGER =
LoggerFactory.getLogger(PersistenceWithJPAServlet.class);
    private DataSource ds;
    private EntityManagerFactory emf;
    /** {@inheritDoc} */
    @SuppressWarnings({ "rawtypes", "unchecked" })
    @Override
    public void init() throws ServletException {
        Connection connection = null;
        try {
            InitialContext ctx = new InitialContext();
            ds = (DataSource) ctx.lookup("java:comp/env/jdbc/DefaultDB");
            Map properties = new HashMap();
            properties.put(PersistenceUnitProperties.NON_JTA_DATASOURCE, ds);
            emf = Persistence.createEntityManagerFactory("persistence-with-
jpa", properties);
        } catch (NamingException e) {
            throw new ServletException(e);
        }
    }
    /** {@inheritDoc} */
    @Override
    public void destroy() {
        emf.close();
    }
    /** {@inheritDoc} */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.getWriter().println("<p>Persistence with JPA Sample!</p>");
        try {
            appendPersonTable(response);
            appendAddForm(response);
        } catch (Exception e) {
            response.getWriter().println("Persistence operation failed with
reason: " + e.getMessage());
            LOGGER.error("Persistence operation failed", e);
        }
    }
    /** {@inheritDoc} */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException,
            IOException {
        try {
            doAdd(request);
            doGet(request, response);
        } catch (Exception e) {
            response.getWriter().println("Persistence operation failed with
reason: " + e.getMessage());
            LOGGER.error("Persistence operation failed", e);
        }
    }
    private void appendPersonTable(HttpServletResponse response) throws
SQLException, IOException {
        // Append table that lists all persons
        EntityManager em = emf.createEntityManager();
        try {
```

```
            @SuppressWarnings("unchecked")
            List<Person> resultList =
em.createNamedQuery("AllPersons").getResultList();
            response.getWriter().println(
                    "<p><table border=\"1\"><tr><th colspan=\"3\">"
                            + (resultList.isEmpty() ? "" : resultList.size()
+ " ")
                            + "Entries in the Database</th></tr>");
            if (resultList.isEmpty()) {
                response.getWriter().println("<tr><td colspan=\"3\">Database
is empty</td></tr>");
            } else {
                response.getWriter().println("<tr><th>First name</th><th>Last
name</th><th>Id</th></tr>");
            }
            IXSSEncoder xssEncoder = XSSEncoder.getInstance();
            for (Person p : resultList) {
                response.getWriter().println(
                        "<tr><td>" + xssEncoder.encodeHTML(p.getFirstName())
+ "</td><td>"
                                + xssEncoder.encodeHTML(p.getLastName()) + "</
td><td>" + p.getId() + "</td></tr>");
            }
            response.getWriter().println("</table></p>");
        } finally {
            em.close();
        }
    }
    private void appendAddForm(HttpServletResponse response) throws
IOException {
        // Append form through which new persons can be added
        response.getWriter().println(
                "<p><form action=\"\" method=\"post\">" + "First name:<input
type=\"text\" name=\"FirstName\">"
                        + " Last name:<input type=\"text\" name=
\"LastName\">"
                        + " <input type=\"submit\" value=\"Add Person
\">" + "</form></p>");
    }
    private void doAdd(HttpServletRequest request) throws ServletException,
IOException, SQLException {
        // Extract name of person to be added from request
        String firstName = request.getParameter("FirstName");
        String lastName = request.getParameter("LastName");
        // Add person if name is not null/empty
        EntityManager em = emf.createEntityManager();
        try {
            if (firstName != null && lastName != null &&
firstName.trim().isEmpty() && !lastName.trim().isEmpty()) {
                Person person = new Person();
                person.setFirstName(firstName);
                person.setLastName(lastName);
                em.getTransaction().begin();
                em.persist(person);
                em.getTransaction().commit();
            }
        } finally {
            em.close();
        }
    }
}
```

4.

4.2.8.2.2.6 Test the Web Application on the Local Server

Test your application by deploying it locally.

Procedure

1. To test your web application on the local server, follow the steps for deploying a web application locally as described in .

   You should see the following output:

2. Enter a first name (for example, `John`) and a last name (for example, `Smith`) and choose *Add Person*.

   `John Smith` is added to the database as shown below:

> **i Note**
>
> If you add more names to the database, they are also listed in the table. This confirms that you have successfully enabled persistence using the `Person` entity.

4.2.8.2.2.7 Deploy Applications Using Persistence on the Cloud from Eclipse

To test your web application in the cloud, define a server in Eclipse.

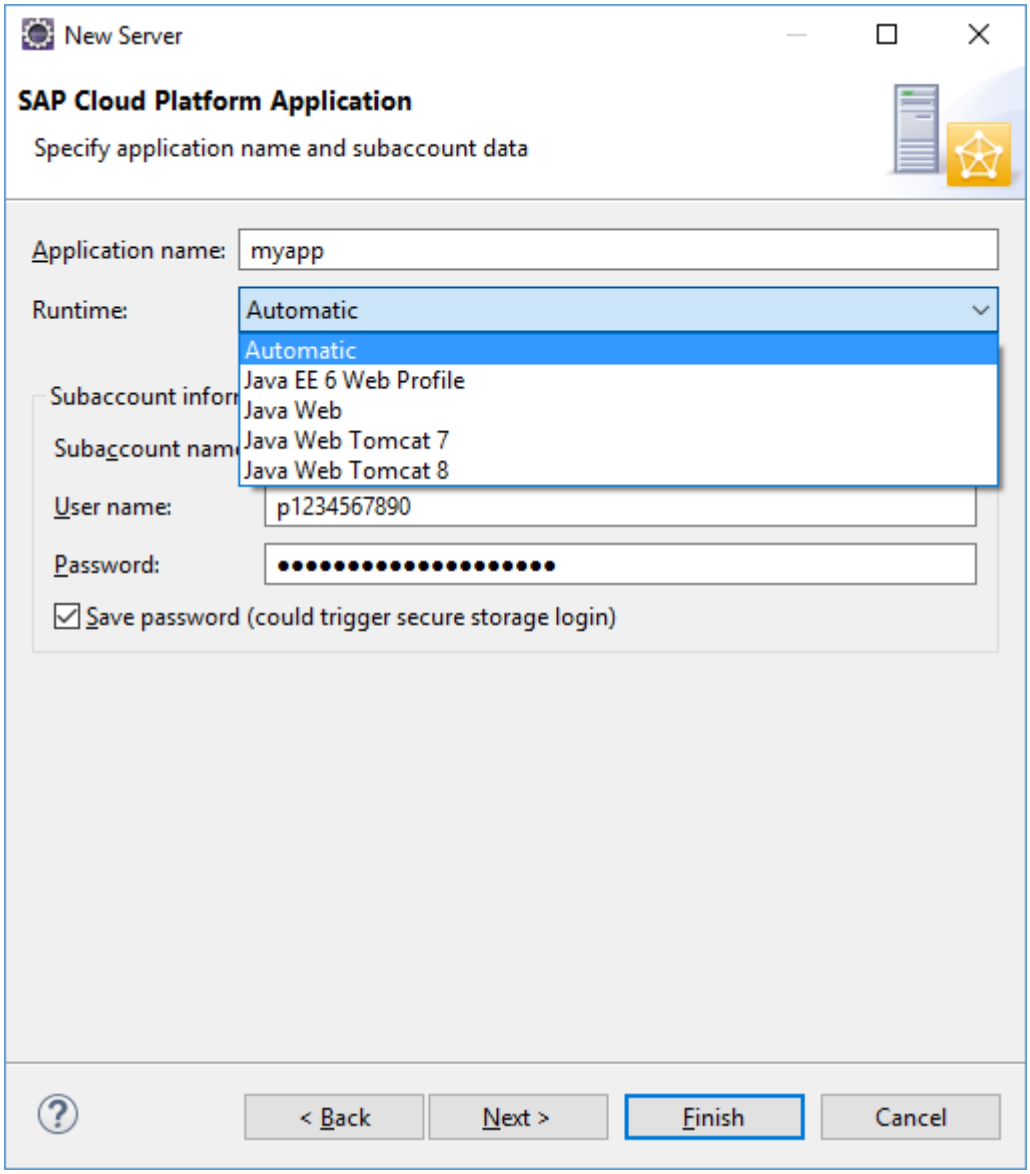
Procedure

1. In the Eclipse IDE, switch to the *Servers* view.
2. Open the context menu define a server with the following settings:
3. Select the server type *SAP* > *SAP Cloud Platform*.
4. Use the host depending on your global account type, then choose *Next*. For more information, see Regions.
5. Specify a unique and identifiable application name.

   Only lowercase Latin letters and digits are allowed.
6. Select a runtime.

   If you leave the default `Automatic` option, the server loads the target runtime of your application.
7. Enter your subaccount name, e-mail or user name, and password, then choose *Next*.
   - If you have previously entered a subaccount and user name for your host, you can select these names from lists.
   - Previously entered hosts also appear in a dropdown list.
   - Select *Save password* to remember and store the password for a given user.

> ⚠ Caution
>
> Do not select your application on the *Add and Remove* screen. Adding an application automatically starts it with the effect that it will fail because no data source binding exists. You will add an application in a later step.



8. Choose *Finish*.
9. In the *Servers* view, open the context menu for the server you just created and choose *Show In* > *Cockpit*.

   The cockpit opens inside Eclipse.

4.2.8.2.2.8 Configure Applications Using the Cockpit

Use the cockpit to create a default binding for your application.

Procedure

1. In the cockpit, select a subaccount, then choose *SAP HANA / SAP ASE* > *Databases & Schemas* in the navigation area.
2. Select the database you want to create a binding for.
3. Choose *Data Source Bindings*. For more information, see .
4. Define a binding (<DEFAULT>) for the application and select a database ID. Choose *Save*.

   You can use an existing database or create a new one.

4.2.8.2.2.9 Start Applications Using Eclipse

Add the application to the new server and start it to deploy the application on the cloud.

Context

> **i Note**
>
> You cannot deploy multiple applications on the same application process. Deployment of a second application on the same application process overwrites any previous deployments. To deploy several applications, deploy each of them on a separate application process.

Procedure

1. On the *Servers* view in Eclipse, open the context menu for the server choose *Add and Remove...* > *<application name>*.
2. To add the application to the server, add the application to the panel on the right side.
3. Choose *Finish*.
4. Start the server.

   This deploys the application and starts it on the SAP BTP.
5. Access the application by clicking the application URL on the application overview page in the cockpit.

   You should see the same output as when the application was tested on the local server.

4.2.8.2.3 Special Settings for EclipseLink Versions Earlier than 2.5

EclipseLink versions prior to 2.5 do not include the SAP HANA database platform. To deploy applications on the SAP HANA database, you need to specify it as the target database and, for application-managed persistence, import the corresponding JAR file into your project.

Container-Managed Persistence

Specify the target database as a persistence unit property:

1. Select the `<project>/Java Resources/src/META-INF/persistence.xml` file and from the context menu, choose *Open With* > *Persistence XML Editor*.
2. On the *Options* tab, enter **`com.sap.persistence.platform.database.HDBPlatform`** in the *Target database* field.

The source code should now contain the following:

```
<properties>
            <property name="eclipselink.target-database"
value="com.sap.persistence.platform.database.HDBPlatform"/>
</properties>
```

Application-Managed Persistence

Specify the target database as shown above or directly in the servlet code, as shown in the example below:

```
ds = (DataSource) ctx.lookup("java:comp/env/jdbc/DefaultDB");
connection = ds.getConnection();
Map properties = new HashMap();
properties.put(PersistenceUnitProperties.NON_JTA_DATASOURCE, ds);
properties.put("eclipselink.target-database",
"com.sap.persistence.platform.database.HDBPlatform");
```

Add the SAP HANA JAR to the Web application project:

1. Select the `<project>/WebContent/WEB-INF/lib` node.
2. From the context menu, choose *Import* > *General* > *File System*, then choose *Next*.
3. Browse to the local directory where you downloaded and unpacked the SAP BTP SDK, select the `repository/plugins` directory, and choose *OK*.
4. Select *com.sap.core.persistence.osgi.hdb.platform_x.y.z.jar* and choose *Finish*.

General Points

Set the target database property before you deploy the application on the SAP HANA database. If you don't, you'll get an error, and if this happens, you need to re-create the table with the correct definitions, setting the

DDL generation type to **`Drop and Create Tables`** , then redeploy the application. Afterwards, set the generation type back to **`Create Tables`** so that you do not lose your data once you deploy again.

When testing the application locally, remove the DDL generation type altogether.

4.2.8.2.4 Persistence Units

A JPA model contains a persistence configuration file, `persistence.xml`, which describes the defined persistence units. A persistence unit in turn defines all entity classes managed by the entity managers in your application and includes the metadata for mapping the entity classes to the database entities.

JPA Provider

The `persistence.xml` file is located in the `META-INF` folder within the persistence unit `src` folder. The JPA persistence provider used by the is org.eclipse.persistence.jpa.PersistenceProvider.

Example

In the `persistence.xml` file in the tutorial *Adding Container-Managed Persistence with JPA (SDK for Java EE 6 Web Profile)*, the persistence unit is named `persistence-with-ejb`, the transaction type is JTA (default setting), and the DDL generation type has been set to `Create Tables`, as shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://
java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
    <persistence-unit name="persistence-with-ejb">
        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
        <class>com.sap.cloud.sample.persistence.Person</class>
        <properties>
            <property name="eclipselink.ddl-generation" value="create-tables" />
        </properties>
    </persistence-unit>
</persistence>
```

DDL Generation Type

The the EclipseLink capabilities to generate database tables. The following values are valid for generating the DDL for the entity specified in the `persistence.xml` file:

- **`none`**: EclipseLink does not generate a DDL (no schema is generated).
- **`create-tables`**: EclipseLink attempts to execute a CREATE TABLE SQL statement. It creates a DDL for non-existent tables and leaves existing tables unchanged (they are not dropped or replaced).

- **`drop-and-create-tables`**: EclipseLink attempts to DROP all tables and then CREATE all tables.

  > **i** Note
  >
  > Drop-and-create tables are often used during the development phase, when there are frequent changes to the schema or data needs to be deleted. Don't forget to change it to **`create-tables`** before you deploy the application; all data is lost when you drop a table.

Transaction Type

JTA transactions are used for container-managed persistence, and resource-local transactions for application-managed persistence. The SDK for Java Web supports resource-local transactions only.

4.2.8.2.5 Using Container-Managed Persistence

Container-managed entity managers are the model most commonly used by Web applications. Container-managed entity managers require JTA transactions and are generally used with stateless session beans and transaction-scoped persistence contexts, which are threadsafe.

Context

The scenario described in this section is based on the Java EE 6 Web Profile runtime. You use a stateless EJB session bean into which the entity manager is injected using the `@PersistenceContext` annotation.

Procedure

1. Configure the persistence units in the `persistence.xml` file to use JTA data sources and JTA transactions.
2. Inject the entity manager into an EJB session bean using the `@PersistenceContext` annotation.

Related Information

Configure Persistence Units [page 948]
Inject Entity Managers [page 949]

4.2.8.2.5.1 Configure Persistence Units

To use container-managed entity managers, configure JTA data sources in the `persistence.xml` file. JTA data sources are managed data sources and are associated with JTA transactions.

Context

To configure JTA data sources, set the transaction type attribute (`transaction-type`) to JTA and specify the names of the JTA data sources (`jta-data-source`), unless the application is using the default data source.

Procedure

1. In the *Project Explorer* view, select `<project>/Java Resources/src/META-INF/persistence.xml`, and from the context menu choose *Open With* > *Persistence XML Editor*.
2. On the *Connection* tab, enter the transaction type **JTA** or leave the default setting, which is JTA.
3. If the application is using an explicitly named data source, enter the JNDI name of the data source in the *JTA data source* field.

   If it uses the default data source, you don't need to specify a data source in the `persistence.xml` file.
4. If the application uses more than one data source, define a corresponding number of persistence units, each with its own data source. The data source name is the JNDI name used for the lookup and must match the name used for the schema binding. To do this, switch to the *Source* tab of the `persistence.xml` file.

   The example below shows the persistence units defined for two data sources, where each data source is associated with a different database:

```
<persistence>
<persistence-unit name="hanadb" transaction-type="JTA">
<provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
<jta-data-source>jdbc/hanaDB</jta-data-source>
        <class>com.sap.cloud.sample.persistence.Person</class>
        <properties>
            <property name="eclipselink.ddl-generation" value="create-
tables" />
        </properties>
</persistence-unit>
<persistence-unit name="maxdb" transaction-type="JTA">
        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
        <jta-data-source>jdbc/maxDB</jta-data-source>
        <class>com.sap.cloud.sample.persistence.Person</class>
        <properties>
            <property name="eclipselink.ddl-generation" value="create-
tables" />
        </properties>
    </persistence-unit>
</persistence>
```

5. Save the file.

Related Information

Configure Data Sources As Connection Properties [page 978]

4.2.8.2.5.2 Inject Entity Managers

EJB session beans, which typically perform the database operations, can use the `@PersistenceContext` annotation to directly inject the entity manager. The corresponding entity manager factory is created transparently by the container.

Context

Procedure

1. In the EJB session bean, inject the entity manager as follows. A persistence context type has not been explicitly specified in the example below and is therefore, by default, transaction-scoped:

   ```
   @PersistenceContext
   private EntityManager em;
   ```

   To use an extended persistence context, set the value of the persistence context type to EXTENDED (@PersistenceContext(type=PersistenceContextType.EXTENDED)), and declare the session bean as stateful. An extended persistence context allows a session bean to maintain its state across multiple JTA transactions. An extended persistence context is not threadsafe.
2. If you have more than one persistence unit, inject the required number of entity managers by specifying the persistence unit name as defined in the `persistence.xml` file:

   ```
   @PersistenceContext(unitName="hanadb")
   private EntityManager em1;
   ...
   @PersistenceContext(unitName="maxdb")
   private EntityManager em2;
   ```
3. Inject an instance of the EJB session bean class into, for example, the servlet of the web application with an annotation in the following form, where **`PersonBean`** is an example session bean class:

   ```
   @EJB PersonBean personBean;
   ```

   The persistence context made available is based on JTA and provides automatic transaction management. Each EJB business method automatically has a managed transaction, unless specified otherwise. The entity manager life cycle, such as instantiation and closing, is controlled by the container. Therefore, do not use methods designed for resource-local transactions, such as `em.getTransaction().begin()`, `em.getTransaction().commit()`, and `em.close()`.

Related Information

Configure Data Sources As Connection Properties [page 978]

4.2.8.2.6 Using Application-Managed Persistence

Application-managed entity managers are created manually using the `EntityManagerFactory` interface. Application-managed entity managers require resource-local transactions and non-JTA data sources, which you must declare as JNDI resource references.

Context

The scenario described in this section is based on the Java Web runtime, which supports only manual creation of the entity manager factory.

Procedure

1. Declare a JNDI resource reference.
2. Configure the persistence units in the `persistence.xml` file to use resource-local transactions and non-JTA data sources.
3. Use a JNDI lookup in the application code to retrieve the data source.
4. Create an entity manager factory and entity manager.

Related Information

Declare JNDI Resource References [page 951]
Configure Persistence Units [page 952]
Retrieve Data Sources [page 952]
Create Entity Managers [page 953]
Apply Dynamic Data Source Lookup [page 955]
Entity Transaction API [page 955]

4.2.8.2.6.1 Declare JNDI Resource References

An application can use one or more data sources. A data source can be a default data source or an explicitly named data source. Before a data source can be used, you must declare it as a JNDI resource reference in the `web.xml` deployment descriptor.

Context

Procedure

1. In the *Project Explorer* view, open the `WebContent/WEB-IN/web.xml` file.
2. Add the following code after the `<servlet-mapping>` elements. The resource reference name is only an example; the one you use may differ.

   ```
   <resource-ref>
         <res-ref-name>jdbc/DefaultDB</res-ref-name>
         <res-type>javax.sql.DataSource</res-type>
   </resource-ref>
   ```

   The resource attributes denote the following:
   - Name – the JNDI name of the resource. The Java EE Specification recommends that you declare the data source reference in the jdbc subcontext (`jdbc/NAME`).
   - Type – the type of resource that is returned during the lookup.
3. If the application uses multiple data sources, add a resource reference for each data source:

   ```
   <resource-ref>
         <res-ref-name>jdbc/datasource1</res-ref-name>
         <res-type>javax.sql.DataSource</res-type>
   </resource-ref>
   <resource-ref>
         <res-ref-name>jdbc/datasource2</res-ref-name>
         <res-type>javax.sql.DataSource</res-type>
   </resource-ref>
   ```

   - The data source name is the JNDI name used for the lookup.
   - The same name must be used for the schema binding.

   > **i Note**
   >
   > If you declare the data source reference in a jdbc subcontext, you must use the same pattern for the name of the schema binding (`jdbc/NAME`).
4. Save the file.

Related Information

Create Schemas Using the Cockpit
Administering Database Schemas

4.2.8.2.6.2 Configure Persistence Units

To use application-managed entity managers, configure resource-local transactions in the `persistence.xml` file. Resource-local transactions are associated with non-JTA data sources (that is, unmanaged data sources) and are explicitly controlled by the application through the `EntityTransaction` interface of the entity manager.

Context

To use resource-local transactions, the transaction type attribute has to be set to RESOURCE_LOCAL, indicating that the entity manager factory should provide resource-local entity managers. When you work with a non-JTA data source, the non-JTA data source element also has to be set in the persistence unit properties in the application code.

Procedure

1. In the *Project Explorer* view, select `<project>/Java Resources/src/META-INF/persistence.xml`, and from the context menu choose *Open With* > *Persistence XML Editor*.
2. On the *Connection* tab, enter the transaction type **`Resource Local`**.
3. Save the file.

4.2.8.2.6.3 Retrieve Data Sources

In the application code, obtain an initial JNDI context by creating a `javax.naming.InitialContext` object, then retrieve the data source by looking up the naming environment through the `InitialContext`. Alternatively, you can directly inject the data source.

Context

Procedure

1. To create an initial JNDI context and look up the data source, add the following code to your application and make sure that the JNDI name matches the one specified in the `web.xml` file:

   ```
   InitialContext ctx = new InitialContext();
   DataSource ds = (DataSource) ctx.lookup("java:comp/env/jdbc/DefaultDB");
   ```

   According to the Java EE Specification, you should add the prefix `java:comp/env` to the JNDI resource name (as specified in the `web.xml`) to form the lookup name. For more information about defining and referencing resources according to the Java EE standard, see the *Java EE Specification*.
2. Alternatively, to directly inject the data source, use the `@Resource` annotation:
   - Default data source
     Since the default data source is provided automatically, it can be injected without an explicit resource name, as shown below. You don't need to also declare the JNDI resource reference in the `web.xml` or `persistence.xml` file:

     ```
     @Resource
     private javax.sql.DataSource ds;
     ```
   - Explicitly named data sources
     These are injected with a specific resource name, as specified in the `web.xml` or `persistence.xml` file:

     ```
     @Resource(name="jdbc/datasource1")
         private javax.sql.DataSource ds1;
     @Resource(name="jdbc/datasource2")
         private javax.sql.DataSource ds2;
     ```

Related Information

Java EE Specification

4.2.8.2.6.4 Create Entity Managers

Use the `EntityManagerFactory` interface to manually create and manage entity managers in your Web application.

Context

Procedure

1. Use `javax.persistence.Persistence.createEntityManagerFactory` to create an `EntityManagerFactory` object that operates on the data source that you have retrieved:

   ```
   Map properties = new HashMap();
   properties.put(PersistenceUnitProperties.NON_JTA_DATASOURCE, ds);
   emf = Persistence.createEntityManagerFactory("persistence-with-jpa",
   properties);
   ```

   In the code above, the non-JTA data source element has been set in the persistence unit properties, and the persistence unit name is the name of the persistence unit as it is declared in the `persistence.xml` file.

   > **i** Note
   >
   > Include the above code in the servlet `init()` method, as illustrated in the tutorial *Adding Application-Managed Persistence with JPA (SDK for Java Web)*, since this method is called only once during initialization when the servlet instance is loaded.

2. Use the entity manager factory obtained above to create an entity manager as follows:

   ```
   EntityManager em = emf.createEntityManager();
   ```

Next Steps

Application-managed entity managers are always extended and therefore retain the entities beyond the scope of a transaction. You should therefore close an entity manager when it is no longer needed by calling `EntityManager.close()`, or alternatively `EntityManager.clear()` wherever appropriate, such as at the end of a transaction. An entity manager cannot be used concurrently by multiple threads, so design your entity manager handling to avoid doing this.

Related Information

Entity Transaction API [page 955]
Configure Data Sources As Connection Properties [page 978]
Tutorial: Adding Application-Managed Persistence with JPA (SDK for Java Web) [page 931]

4.2.8.2.6.5 Entity Transaction API

When working with a resource-local entity manager, use the EntityTransaction API to manually set the transaction boundaries in your application code. You can obtain the entity transaction attached to the entity manager by calling `EntityManager.getTransaction()`.

To create and update data in the database, you need an active transaction. The EntityTransaction API provides the `begin()` method for starting a transaction, and the `commit()` and `rollback()` methods for ending a transaction. When a commit is executed, all changes are synchronized with the database.

Example

The tutorial code (*Adding Application-Managed Persistence with JPA (SDK for Java Web)*) shows how to create and persist an entity:

```
Person person = new Person("<name>");
em.getTransaction().begin();
em.persist(person);
em.getTransaction().commit();
em.close();
```

The `EntityManager.persist()` method makes an entity persistent by associating it with an entity manager. It is inserted into the database when the `commit()` method is called. The `persist()` method can be called only on new entities.

Related Information

Tutorial: Adding Application-Managed Persistence with JPA (SDK for Java Web) [page 931]

4.2.8.2.6.6 Apply Dynamic Data Source Lookup

The data source is determined dynamically at runtime and does not need to be defined in the `web.xml` or `persistence.xml` file. This allows you to bind additional schemas to an application and obtain the corresponding data source, without having to modify the application code or redeploy the application.

Context

A dynamic JNDI lookup is applied as follows, depending on whether you are using an unmanaged or a managed data source:

- Unmanaged - supported in all Java runtimes

```
context.lookup("unmanageddatasource:<data source name>")
```

- Managed - supported in Java EE runtimes (Java EE 6 Web Profile and Java EE 7 Web Profile TomEE 7)

```
context.lookup("manageddatasource:<data source name>")
```

The steps described below are based on JPA application-managed persistence using a Java runtime.

Procedure

1. Create the persistence unit to be used for the dynamic data source lookup:
   a. In the *Project Explorer* view, select `<project>/Java Resources/src/META-INF/persistence.xml`, and from the context menu choose *Open With* > *Persistence XML Editor*.
   b. Switch to the *Source* tab of the `persistence.xml` file and create a persistence unit, as shown in the example below. The corresponding data source is not defined in either the `persistence.xml` or `web.xml` file:

```
<persistence-unit name="mypersistenceunit" transaction-
type="RESOURCE_LOCAL">
<provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
<class>com.sap.cloud.sample.persistence.Person</class>
<properties>
<property name="eclipselink.ddl-generation" value="create-tables"/>
</properties>
</persistence-unit>
```

2. In the servlet code, implement a JNDI data source lookup. In the example below, the data source name is "mydatasource":

```
ds = (DataSource) context.lookup("unmanageddatasource:mydatasource");
```

3. Create an entity manager factory in the normal manner. In the example below, the persistence unit is named "mypersistenceunit", as defined in the `persistence.xml` file:

```
Map properties = new HashMap();
            properties.put(PersistenceUnitProperties.NON_JTA_DATASOURCE, ds);
emf = Persistence.createEntityManagerFactory("mypersistenceunit", properties);
```

4. Use the console client to create a schema binding with the same data source name. To do this, open the command window in the `<SDK>/tools` folder and enter the bind-schema [page 1370] command, using the data source name you defined in step 2:

```
neo bind-schema -h <host> -u <user> -a <subaccount> -b <application name> --
data-source mydatasource --id <schema ID>
```

> **i Note**
>
> Note that you need to use the same data source name you have defined in step 2.

4.2.8.2.7 Entity Classes

To declare a class as an entity and define how that entity maps to the relevant database table, you can either decorate the Java object with metadata using Java annotations or denote it as an entity in the XML descriptor.

The Dali Java Persistence Tools, which are provided as part of the Eclipse IDE for Java EE Developers, allow you to use a JPA diagram editor to create, edit, and display entities and their relationships (your application's data model) in a graphical environment.

Example

The tutorial *Adding Application-Managed Persistence with JPA (SDK for Java Web)* defines the entity class `Person`, as shown in the following:

```
package com.sap.cloud.sample.persistence;
import javax.persistence.*;
@Entity
@Table(name = "T_PERSON")
@NamedQuery(name = "AllPersons", query = "select p from Person p")
public class Person {
    @Id
    @GeneratedValue
    private long id;
    @Basic
    private String FirstName;
    @Basic
    private String LastName;
```

- The **`Person`** class has been annotated as an entity: @Entity.
- The @Table annotation maps the entity to the database table **`T_PERSON`**.
- The @NamedQuery annotation indicates that a static query has been created in the metadata. The name element of @NamedQuery gives the name of the query that will be used with the `createNamedQuery()` method, while the query element specifies the actual query.
- The definition of the field serving as the unique identifier of the entity has been annotated with @Id, indicating it is the primary key in the database. Its value is generated automatically (@GeneratedValue).

Related Information

Tutorial: Adding Application-Managed Persistence with JPA (SDK for Java Web) [page 931]

Dali Java Persistence Tools User Guide

4.2.8.2.8 Creating SAP HANA Column-Store Tables

The SAP HANA database lets you create tables with row-based storage or column-based storage. By default, tables are created with row-based storage, but you can change the type of table storage you have applied, if necessary.

The example below shows the SQL syntax used by the SAP HANA database to create different table types. The first two SQL statements both create row-store tables, the third a column-store table, and the fourth changes the table type from row-store to column-store:

```
CREATE TABLE T_PERSON
CREATE ROW TABLE T_PERSON
CREATE COLUMN TABLE T_PERSON
ALTER TABLE T_PERSON COLUMN
```

EclipseLink JPA

When using EclipseLink JPA for data persistence, the table type applied by default in the SAP HANA database is row-store. To create a column-store table or alter an existing row-store table, you can manually modify your database using SQL DDL statements, or you can use open source tools, such as Liquibase (with plain SQL statements), to handle automated database migrations.

Due to the limitations of the EclipseLink schema generation feature, you'll need to use one of the above options to handle the life cycle management of your database objects.

ALTER TABLE Example

You can use the ALTER TABLE statement to change a row-store table in the SAP HANA database to a column-store table. The example is based on the *Adding Application-Managed Persistence with JPA (SDK for Java Web)* tutorial and has been designed specifically for this tutorial and use case.

The example allows you to take advantage of the automatic table generation feature provided by JPA EclipseLink. You merely alter the existing table at an appropriate point, when the schema containing the relevant table has just been created. The applicable code snippet is added to the `init()` method of the servlet (`PersistenceWithJPAServlet`). The main changes to the servlet code are outlined below:

1. Since the table must already exist when the ALTER statement is called, a small workaround has been introduced in the `init()` method. An entity manager is created at an earlier stage than in the original version of the tutorial to trigger the generation of the schema:

   ```
   //workaround: create EntityManager to trigger schema generation
   emf.createEntityManager().close();
   ```

2. The SAP HANA database table SYS.M_TABLES contains information about all row and column tables in the current schema. A new method, which uses this table to check that T_PERSON is not already a column-store table, has been added to the servlet.
3. Another new method alters the table using the SQL statement ALTER TABLE <table name> COLUMN.

To apply the solution, replace the entire servlet class `PersistenceWithJPAServlet` with the following content:

```
package com.sap.cloud.sample.persistence;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;
import org.eclipse.persistence.config.PersistenceUnitProperties;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.sap.security.core.server.csi.IXSSEncoder;
import com.sap.security.core.server.csi.XSSEncoder;
/**
* Servlet implementing a simple JPA based persistence sample application for SAP
BTP.
 */
public class PersistenceWithJPAServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final Logger LOGGER =
LoggerFactory.getLogger(PersistenceWithJPAServlet.class);
    private static final String SQL_GET_TABLE_TYPE = "SELECT TABLE_NAME,
TABLE_TYPE FROM SYS.M_TABLES WHERE TABLE_NAME = ?";
    private static final String PERSON_TABLE_NAME = "T_PERSON";
    private DataSource ds;
    private EntityManagerFactory emf;
    /** {@inheritDoc} */
    @SuppressWarnings({ "rawtypes", "unchecked" })
    @Override
    public void init() throws ServletException {
        Connection connection = null;
        try {
            InitialContext ctx = new InitialContext();
            ds = (DataSource) ctx.lookup("java:comp/env/jdbc/DefaultDB");
            Map properties = new HashMap();
            properties.put(PersistenceUnitProperties.NON_JTA_DATASOURCE, ds);
            boolean onHANA = runsOnHANADatabase();
            if (onHANA) {
                properties.put("eclipselink.target-database",
"com.sap.persistence.platform.database.HDBPlatform");
            }
            emf = Persistence.createEntityManagerFactory("persistence-with-jpa",
properties);
            // convert T_PERSON to column table
            // workaround: create EntityManager to trigger schema generation
            emf.createEntityManager().close();
            if (onHANA) {
                convertToColumnTable(PERSON_TABLE_NAME);
            }
        } catch (NamingException e) {
            throw new ServletException(e);
        } catch (SQLException e) {
            LOGGER.error("Could not determine database product.", e);
```

```
            throw new ServletException(e);
        } finally {
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException x) {
                    LOGGER.debug("Unable to close connection.", x);
                }
            }
        }
    }
    /** {@inheritDoc} */
    @Override
    public void destroy() {
        emf.close();
    }
    /** {@inheritDoc} */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.getWriter().println("<p>Persistence with JPA Sample!</p>");
        try {
            appendPersonTable(response);
            appendAddForm(response);
        } catch (Exception e) {
            response.getWriter().println("Persistence operation failed with
reason: " + e.getMessage());
            LOGGER.error("Persistence operation failed", e);
        }
    }
    /** {@inheritDoc} */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException,
            IOException {
        try {
            doAdd(request);
            doGet(request, response);
        } catch (Exception e) {
            response.getWriter().println("Persistence operation failed with
reason: " + e.getMessage());
            LOGGER.error("Persistence operation failed", e);
        }
    }
    private void appendPersonTable(HttpServletResponse response) throws
SQLException, IOException {
        // Append table that lists all persons
        EntityManager em = emf.createEntityManager();
        try {
            @SuppressWarnings("unchecked")
            List<Person> resultList =
em.createNamedQuery("AllPersons").getResultList();
            response.getWriter().println(
                    "<p><table border=\"1\"><tr><th colspan=\"3\">"
                            + (resultList.isEmpty() ? "" : resultList.size() +
" ")
                            + "Entries in the Database</th></tr>");
            if (resultList.isEmpty()) {
                response.getWriter().println("<tr><td colspan=\"3\">Database is
empty</td></tr>");
            } else {
                response.getWriter().println("<tr><th>First name</th><th>Last
name</th><th>Id</th></tr>");
            }
            IXSSEncoder xssEncoder = XSSEncoder.getInstance();
            for (Person p : resultList) {
                response.getWriter().println(
```

```
                            "<tr><td>" + xssEncoder.encodeHTML(p.getFirstName()) +
"</td><td>"
                                    + xssEncoder.encodeHTML(p.getLastName()) + "</
td><td>" + p.getId() + "</td></tr>");
            }
            response.getWriter().println("</table></p>");
        } finally {
            em.close();
        }
    }
    private void appendAddForm(HttpServletResponse response) throws IOException {
        // Append form through which new persons can be added
        response.getWriter().println(
                "<p><form action=\"\" method=\"post\">" + "First name:<input
type=\"text\" name=\"FirstName\">"
                        + " Last name:<input type=\"text\" name=\"LastName
\">"
                        + " <input type=\"submit\" value=\"Add Person\">" +
"</form></p>");
    }
    private void doAdd(HttpServletRequest request) throws ServletException,
IOException, SQLException {
        // Extract name of person to be added from request
        String firstName = request.getParameter("FirstName");
        String lastName = request.getParameter("LastName");
        // Add person if name is not null/empty
        EntityManager em = emf.createEntityManager();
        try {
            if (firstName != null && lastName != null && !
firstName.trim().isEmpty() && !lastName.trim().isEmpty()) {
                Person person = new Person();
                person.setFirstName(firstName);
                person.setLastName(lastName);
                em.getTransaction().begin();
                em.persist(person);
                em.getTransaction().commit();
            }
        } finally {
            em.close();
        }
    }
    private boolean runsOnHANADatabase() throws SQLException {
        boolean onHANA = false;
        Connection connection = ds.getConnection();
        try {
            String databaseProductName =
connection.getMetaData().getDatabaseProductName();
            if (databaseProductName.equals("HDB")) {
                onHANA = true;
            }
        } finally {
            connection.close();
        }
        return onHANA;
    }
    private void convertToColumnTable(String tableName) throws SQLException {
        if (!isColumnTable(tableName)) {
            Connection connection = ds.getConnection();
            try {
                String sql = "ALTER TABLE " + tableName + " COLUMN";
                PreparedStatement stmt = connection.prepareStatement(sql);
                stmt.executeUpdate();
                stmt.close();
            } finally {
                connection.close();
            }
        }
    }
```

```
    private boolean isColumnTable(String tableName) throws SQLException {
        boolean exists = false;
        boolean columnTable = false;
        Connection connection = ds.getConnection();
        String tableTypeStart = null;
        try {
            PreparedStatement stmt =
connection.prepareStatement(SQL_GET_TABLE_TYPE);
            stmt.setString(1, tableName);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                exists = true;
                tableTypeStart = rs.getString(2);
                break;
            }
            rs.close();
            if (!exists) {
                throw new SQLException("Table " + tableName + " does not exist");
            }
            if (tableTypeStart.equalsIgnoreCase("COLUMN")) {
                columnTable = true;
            }
        } finally {
            connection.close();
        }
        return columnTable;
    }
}
```

Related Information

Tutorial: Adding Application-Managed Persistence with JPA (SDK for Java Web) [page 931]

4.2.8.2.9 EclipseLink Weaving

EclipseLink provides weaving as a means of enhancing JPA entities and classes for performance optimization. At present, SAP BTP supports static weaving only. Static weaving occurs at compile time and is available in both the Java Web and Java EE 6 Web Profile environments.

Dynamic weaving is currently not supported on SAP BTP.

Prerequisites

- For static weaving to work, the entity classes must be listed in the `persistence.xml` file.
- EclipseLink Library:
  To use the EclipseLink weaving options in your web applications, add the EclipseLink library to the classpath:
  - SDK for Java Web
    The EclipseLink library has already been added to the `WebContent/WEB-INF/lib` folder, since it is required for the JPA persistence scenario.

- SDK for Java EE 6 Web Profile
  The EclipseLink library is already part of the SDK for Java EE 6 Web Profile, allowing you to run JPA scenarios without any additional steps. To use the weaving options, however, you add the EclipseLink library to the classpath, as described below.

SDK for Java EE 6 Web Profile: Adding the EclipseLink Library to the Classpath

1. In the Eclipse IDE in the *Project Explorer* view, select the web application and from the context menu choose *Properties*.
2. In the tree, select *JPA*.
3. In the *Platform* section, select the correct EclipseLink version, which should match the version available in the SDK.
4. In the *JPA implementation* section, select the type **`User Library`**.
5. To the right of the user library list box, choose *Download library*.
6. Select the correct version of the EclipseLink library (currently **`EclipseLink 2.5.2`**) and choose *Next*.
7. Accept the EclipseLink license and choose *Finish*.
8. The new user library now appears; make sure it is selected.
9. Unselect *Include libraries with this application* and choose *OK*.

Enabling Static Weaving in Eclipse

1. In the Eclipse IDE in the *Project Explorer* view, select the web application and from the context menu choose *Properties*.
2. In the tree, select *JPA* > *EclipseLink*.
3. In the *Static weaving* section, select *Weave classes on build*.
4. Leave the default values for the source classes, target classes, and persistence XML root; however you may need to adapt them if you have a non-standard web application project layout. Choose *OK*.

   > **i** Note
   >
   > If you change the target class settings, make sure you deploy these classes.

Your web application project is rebuilt so that the JPA entity class files contain weaving information. This also occurs on each (incremental) project build. The woven entity classes will are whenever you publish the web application to the cloud.

More Information

For information about using an ant task or the command line to perform static weaving, see the EclipseLink User Guide.

4.2.8.3 Programming with Plain JDBC

Program with JDBC in the Neo environment in cases in which its low-level control is more appropriate than JPA.

> ⚠ Caution
>
> Creating your own `DataSource` is not supported in the Neo environment and may cause issues. JDBC URLs should only be provided via backend services, not hardcoded, as they may change due to internal updates of the Neo environment.

Working with JDBC entails manually writing SQL statements to read and write objects from and to the database.

JNDI Resource Reference

An application can use one or more data sources. A data source can be a default, or explicitly named. Either way, before a data source can be used, you must declare it as a JNDI resource reference.

Declare a JNDI resource reference to a JDBC data source in the `web.xml` deployment descriptor located in the `WebContent/WEB-INF` directory as shown below. The resource reference name is only an example:

```
<resource-ref>
    <res-ref-name>jdbc/DefaultDB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
</resource-ref>
```

The resource attributes are as follows:

- `<res-ref-name>`: The JNDI name of the resource. The Java EE Specification recommends that you declare the data source reference in the jdbc subcontext (`jdbc/NAME`).
- `<res-type>`: The type of resource that is returned during the lookup.

Add the `<resource-ref>` elements after the `<servlet-mapping>` elements in the deployment descriptor.

If the application uses multiple data sources, add a resource reference for each data source:

```
<resource-ref>
      <res-ref-name>jdbc/datasource1</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
</resource-ref>
<resource-ref>
      <res-ref-name>jdbc/datasource2</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
</resource-ref>
```

Data Source Lookup

You can obtain an initial JNDI context from Tomcat by creating a `javax.naming.InitialContext` object. Then consume the data source by looking up the naming environment through the `InitialContext`, as follows:

```
InitialContext ctx = new InitialContext();
DataSource ds = (DataSource) ctx.lookup("java:comp/env/jdbc/DefaultDB");
```

According to the Java EE Specification, you should add the prefix `java:comp/env` to the JNDI resource name (as specified in `web.xml`) to form the lookup name.

If the application uses multiple data sources, construct the lookup in a similar manner to the following:

```
InitialContext ctx = new InitialContext();
DataSource ds1 = (DataSource) ctx.lookup("java:comp/env/jdbc/datasource1");
DataSource ds2 = (DataSource) ctx.lookup("java:comp/env/jdbc/datasource2");
```

Data Source Injection

You can directly inject the data source using annotations as shown below.

- Default data source
  A default data source is automatically provided; therefore, you don't need to declare the JNDI resource reference in the `web.xml` deployment descriptor or in the resource name. To inject the data source, use the following code:

  ```
  @Resource
  private javax.sql.DataSource ds;
  ```

- If the application uses explicitly named data sources, you must first declare them in the `web.xml` file. Inject them as shown in the following example:

  ```
  @Resource(name="jdbc/datasource1")
      private javax.sql.DataSource ds1;
  @Resource(name="jdbc/datasource2")
      private javax.sql.DataSource ds2;
  ```

JDBC Connection

The data source let you create a JDBC connection to the database. You can use the resulting Connection object to instantiate a Statement object and execute SQL statements, as shown in the following example:

```
private static final String STMT_SELECT_ALL = "SELECT ID, FIRSTNAME, LASTNAME
FROM " + TABLE_NAME;
Connection conn = dataSource.getConnection();
        try {
        PreparedStatement pstmt = conn.prepareStatement(STMT_SELECT_ALL);
        ResultSet rs = pstmt.executeQuery();
  ...
```

Database Tables

Use plain SQL statements to create the tables you require. Since there is currently no tool support available, you have to manually maintain the table life cycles. The exact syntax you'll use may differ, depending on the underlying database. The Connection object provides metadata about the underlying database and its tables and fields, which can be accessed as shown in the code below:

```
String database = conn.getMetaData().getDatabaseProductName();
```

To create a table in the Apache Derby database, you could use the following SQL statement executed with a PreparedStatement object:

```
private static final String STMT_CREATE_TABLE_DERBY = "CREATE TABLE "
        + TABLE_NAME + " (ID INTEGER GENERATED ALWAYS AS IDENTITY
              PRIMARY KEY, " +  "FIRSTNAME VARCHAR (255), LASTNAME VARCHAR
              (255))";
PreparedStatement pstmt = conn.prepareStatement(STMT_CREATE_TABLE_DERBY);
pstmt.executeUpdate();
```

DAO Objects and SQL Statements

See the tutorial *Adding Persistence Using JDBC* for information about executing SQL statements and applying the Data Access Object (DAO) design pattern in your Web application.

Related Information

Tutorial: Adding Persistence with JDBC (SDK for Java Web) [page 966]

Java EE Specification

4.2.8.3.1 Tutorial: Adding Persistence with JDBC (SDK for Java Web)

Use JDBC to persist data in a simple Java EE web application that manages a list of persons.

Prerequisites

- Download and set up your Eclipse IDE, SAP BTP Tools for Java, and SDK for Java Web. For more information, see Setting Up the Development Environment [page 845].

- Create a database. For more information, see Creating Databases.
- Set up your runtime environment in the Eclipse IDE.
- Develop or import a Java Web application in Eclipse IDE. For more information, see Developing Java Applications [page 871].
  The application is also available as a sample in the SAP BTP SDK for Neo environment for Java Web:
  - Sample name: `persistence-with-jdbc`
  - Location: `<sdk>/samples` folder

  For more information, see Using Samples [page 850].

Context

Perform the following steps:

1. Create a Dynamic Web Project and Servlet with JDBC [page 967]
2. Create the Person Entity [page 968]
3. Create the Person DAO [page 969]
4. Prepare the Web Application Project for JDBC [page 971]
5. Extend the Servlet to Use Persistence [page 972]
6. Test the Web Application on the Local Server [page 974]
7. Deploy Applications Using Persistence on the Cloud from Eclipse [page 975]
8. Configure Applications Using the Cockpit [page 977]
9. Start Applications Using Eclipse [page 977]

4.2.8.3.1.1 Create a Dynamic Web Project and Servlet with JDBC

Create a dynamic web project and add a servlet, which you'll extend later in the tutorial.

Procedure

1. From the Eclipse main menu, choose *File* > *New* > *Dynamic Web Project*.
2. Enter the *Project name* **`persistence-with-jdbc`**.
3. In the *Target Runtime* pane, select *Java Web* as the runtime to use to deploy the application.
4. Leave the default values for the other project settings and choose *Next*.
5. On the *Java* screen, leave the default settings and choose *Next*.
6. In the *Web Module* configuration settings, select *Generate web.xml deployment descriptor* and choose *Finish*.
7. To add a servlet to the project you have just created, choose *File* > *New* > *Web* > *Servlet* from the Eclipse main menu.

8. Enter the Java package **com.sap.cloud.sample.persistence** and the class name **PersistenceWithJDBCServlet**.
9. Choose *Finish* to generate the servlet.

4.2.8.3.1.2 Create the Person Entity

Create an entity class named **Person** with basic attributes.

Procedure

1. In the *Project Explorer* view, select the `persistence-with-jdbc/Java Resources/src/com.sap.cloud.sample.persistence` node.
2. From the context menu, choose *New* > *Class*, verify that the package entered is **com.sap.cloud.sample.persistence**, enter the class name **Person**, and choose *Finish*.
3. Open the file in the text editor and insert the following content:

```
package com.sap.cloud.sample.persistence;
/**
 * Class holding information on a person.
 */
public class Person {
    private String id;
    private String firstName;
    private String lastName;
    public String getId() {
        return id;
    }
    public void setId(String newId) {
        this.id = newId;
    }
    public String getFirstName() {
        return this.firstName;
    }
    public void setFirstName(String newFirstName) {
        this.firstName = newFirstName;
    }
    public String getLastName() {
        return this.lastName;
    }
    public void setLastName(String newLastName) {
        this.lastName = newLastName;
    }
}
```

4. Save the class.

4.2.8.3.1.3 Create the Person DAO

Create a DAO class, **`PersonDAO`**, in which you encapsulate the access to the persistence layer.

Procedure

1. In the *Project Explorer* view, select the node.
2. From the context menu, choose *New* > *Class*, verify that the package entered is persistence-with-jdbc/Java Resources/src/com.sap.cloud.sample.persistence**`com.sap.cloud.sample.persistence`**, enter the class name **`PersonDAO`**, and choose *Finish*.
3. Open the file in the text editor and insert the following content:

```
package com.sap.cloud.sample.persistence;
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import javax.sql.DataSource;
/**
 * Data access object encapsulating all JDBC operations for a person.
 */
public class PersonDAO {
    private DataSource dataSource;
    /**
     * Create new data access object with data source.
     */
    public PersonDAO(DataSource newDataSource) throws SQLException {
        setDataSource(newDataSource);
    }
    /**
     * Get data source which is used for the database operations.
     */
    public DataSource getDataSource() {
        return dataSource;
    }
    /**
     * Set data source to be used for the database operations.
     */
    public void setDataSource(DataSource newDataSource) throws SQLException {
        this.dataSource = newDataSource;
        checkTable();
    }
    /**
     * Add a person to the table.
     */
    public void addPerson(Person person) throws SQLException {
        Connection connection = dataSource.getConnection();
        try {
            PreparedStatement pstmt = connection
                    .prepareStatement("INSERT INTO PERSONS (ID, FIRSTNAME,
LASTNAME) VALUES (?, ?, ?)");
            pstmt.setString(1, UUID.randomUUID().toString());
            pstmt.setString(2, person.getFirstName());
            pstmt.setString(3, person.getLastName());
            pstmt.executeUpdate();
```

```
        } finally {
            if (connection != null) {
                connection.close();
            }
        }
    }
    /**
     * Get all persons from the table.
     */
    public List<Person> selectAllPersons() throws SQLException {
        Connection connection = dataSource.getConnection();
        try {
            PreparedStatement pstmt = connection
                    .prepareStatement("SELECT ID, FIRSTNAME, LASTNAME FROM
PERSONS");
            ResultSet rs = pstmt.executeQuery();
            ArrayList<Person> list = new ArrayList<Person>();
            while (rs.next()) {
                Person p = new Person();
                p.setId(rs.getString(1));
                p.setFirstName(rs.getString(2));
                p.setLastName(rs.getString(3));
                list.add(p);
            }
            return list;
        } finally {
            if (connection != null) {
                connection.close();
            }
        }
    }
    /**
     * Check if the person table already exists and create it if not.
     */
    private void checkTable() throws SQLException {
        Connection connection = null;
        try {
            connection = dataSource.getConnection();
            if (!existsTable(connection)) {
                createTable(connection);
            }
        } finally {
            if (connection != null) {
                connection.close();
            }
        }
    }
    /**
     * Check if the person table already exists.
     */
    private boolean existsTable(Connection conn) throws SQLException {
        DatabaseMetaData meta = conn.getMetaData();
        ResultSet rs = meta.getTables(null, null, "PERSONS", null);
        while (rs.next()) {
            String name = rs.getString("TABLE_NAME");
            if (name.equals("PERSONS")) {
                return true;
            }
        }
        return false;
    }
    /**
     * Create the person table.
     */
    private void createTable(Connection connection) throws SQLException {
        PreparedStatement pstmt = connection
                .prepareStatement("CREATE TABLE PERSONS "
                        + "(ID VARCHAR(255) PRIMARY KEY, "
```

```
                        + "FIRSTNAME VARCHAR (255),"
                        + "LASTNAME VARCHAR (255))");
            pstmt.executeUpdate();
        }
    }
```

4. Save the class.

4.2.8.3.1.4 Prepare the Web Application Project for JDBC

Prepare the web application project by adding the XSS Protection Library, adapting the Java build path order, and adding the resource reference description to the `web.xml` file.

Procedure

1. Add the XSS Protection Library to the web application project:
   a. In the *persistence-with-jdbc/JavaProject Explorer* view, select the `persistence-with-jdbc/WebContent/WEB-INF/lib` node.
   b. From the context menu, choose *Import* > *General* > *File System*, then choose *Next*.
   c. Browse to the local directory where you downloaded and unpacked the SAP BTP SDK, select the `repository/plugins` directory, and choose *OK*.
2. Adapt the Java build path order:
   a. In the *Project Explorer* view, select the `persistence-with-jdbc` node, and from the context menu choose *Properties*.
   b. Select *Java Build Path* and switch to the *Order and Export* tab.
   c. Select `Web App Libraries`.
   d. Choose *OK*.
3. Add the resource reference description to `web.xml`:
   a. In the *Project Explorer* view, expand the `persistence-with-jdbc/WebContent/WEB-INF` node.
   b. Select `web.xml`, and from the context menu choose *Open With* > *Text Editor*.
   c. Insert the following content after the `<servlet-mapping>` or `<welcome-file-list>` element:

```
<resource-ref>
    <res-ref-name>jdbc/DefaultDB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
</resource-ref>
```

   d. Save the file.
4. **Optional:** Modify the servlet deployment descriptor information:
   a. In the `web.xml` file, replace the URL pattern "/PersistenceWithJDBCServlet" that was generated for the servlet with "/" as shown below:

```
<servlet-mapping>
    <servlet-name>PersistenceWithJDBCServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

> **Note**
>
> If your servlet version is 3.0 or higher, simply change the *WebServlet* annotation in the `PersistenceWithJDBCServlet.java` class to `@WebServlet("/")`.

b. Save the file.

> **Note**
>
> An application's URL path contains the context root followed by the optional URL pattern ("/<URL pattern>"). The servlet URL pattern that is automatically generated by Eclipse uses the servlet's class name as part of the pattern. Since the cockpit shows only the context root, this means that you cannot directly open the application in the cockpit without adding the servlet name. To call the application by only the context root, use "/" as the URL mapping, then you will no longer have to correct the URL in the browser.

4.2.8.3.1.5 Extend the Servlet to Use Persistence

The exended servlet adds **`Person`** entity objects to the database, retrieves their details, and displays them on the screen.

Procedure

1. In the *Project Explorer* view, expand the `persistence-with-jdbc/src/com.sap.cloud.sample.persistence` node.
2. Select `PersistenceWithJDBCServlet.java`, and from the context menu choose *Open With* > *Java Editor*.
3. In the opened editor, replace the entire servlet class with the following content:

```
package com.sap.cloud.sample.persistence;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.sap.security.core.server.csi.IXSSEncoder;
import com.sap.security.core.server.csi.XSSEncoder;
/**
 * Servlet implementing a simple JDBC based persistence sample application for
 * SAP BTP.
 */
public class PersistenceWithJDBCServlet extends HttpServlet {
```

```
    private static final long serialVersionUID = 1L;
    private static final Logger LOGGER = LoggerFactory
            .getLogger(PersistenceWithJDBCServlet.class);
    private PersonDAO personDAO;
    /** {@inheritDoc} */
    @Override
    public void init() throws ServletException {
        try {
            InitialContext ctx = new InitialContext();
            DataSource ds = (DataSource) ctx
                    .lookup("java:comp/env/jdbc/DefaultDB");
            personDAO = new PersonDAO(ds);
        } catch (SQLException e) {
            throw new ServletException(e);
        } catch (NamingException e) {
            throw new ServletException(e);
        }
    }
    /** {@inheritDoc} */
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        response.getWriter().println("<p>Persistence with JDBC!</p>");
        try {
            appendPersonTable(response);
            appendAddForm(response);
        } catch (Exception e) {
            response.getWriter().println(
                    "Persistence operation failed with reason: "
                            + e.getMessage());
            LOGGER.error("Persistence operation failed", e);
        }
    }
    /** {@inheritDoc} */
    @Override
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        try {
            doAdd(request);
            doGet(request, response);
        } catch (Exception e) {
            response.getWriter().println(
                    "Persistence operation failed with reason: "
                            + e.getMessage());
            LOGGER.error("Persistence operation failed", e);
        }
    }
    private void appendPersonTable(HttpServletResponse response)
                throws SQLException, IOException {
            // Append table that lists all persons
            List<Person> resultList = personDAO.selectAllPersons();
            response.getWriter().println(
                        "<p><table border=\"1\"><tr><th colspan=\"3\">"
                        + resultList.size() + " entries in the Database</th></
tr>" +
                        (resultList.isEmpty()
                        ? "<tr><td colspan=\"3\">Database is empty</td></tr>"
                        : "<tr><th>First Name</th><th>Last Name</th><th>Id</
th></tr>"));
            IXSSEncoder xssEncoder = XSSEncoder.getInstance();
            for (Person p : resultList) {
                  response.getWriter().println(
                              "<tr><td>" +
xssEncoder.encodeHTML(p.getFirstName())
                                          + "</td><td>"
```

```
                                                +
xssEncoder.encodeHTML(p.getLastName())
                                                + "</td><td>" + p.getId() + "</td></
tr>");
            }
            response.getWriter().println("</table></p>");
     }
    private void appendAddForm(HttpServletResponse response) throws
IOException {
        // Append form through which new persons can be added
        response.getWriter()
                .println(
                        "<p><form action=\"\" method=\"post\">"
                                + "First name:<input type=\"text\" name=
\"FirstName\">"
                                + " Last name:<input type=\"text\" name=
\"LastName\">"
                                + " <input type=\"submit\" value=\"Add
Person\">"
                                + "</form></p>");
    }
    private void doAdd(HttpServletRequest request) throws ServletException,
            IOException, SQLException {
        // Extract name of person to be added from request
        String firstName = request.getParameter("FirstName");
        String lastName = request.getParameter("LastName");
        // Add person if name is not null/empty
        if (firstName != null && lastName != null
                && !firstName.trim().isEmpty() && !lastName.trim().isEmpty())
{
            Person person = new Person();
            person.setFirstName(firstName.trim());
            person.setLastName(lastName.trim());
            personDAO.addPerson(person);
        }
    }
}
```

4. Save the servlet. The project should compile without any errors.

4.2.8.3.1.6 Test the Web Application on the Local Server

Test your application by deploying it locally.

Procedure

1. To test your web application on the local server, follow the steps for deploying a web application locally as described in .

   You should see the following output:



2. Enter a first name (for example, `John`) and a last name (for example, `Smith`) and choose *Add Person*.

   `John Smith` is added to the database as shown below:



> ### i Note
>
> If you add more names to the database, they are also listed in the table. This confirms that you have successfully enabled persistence using the `Person` entity.

4.2.8.3.1.7 Deploy Applications Using Persistence on the Cloud from Eclipse

To test your web application in the cloud, define a server in Eclipse.

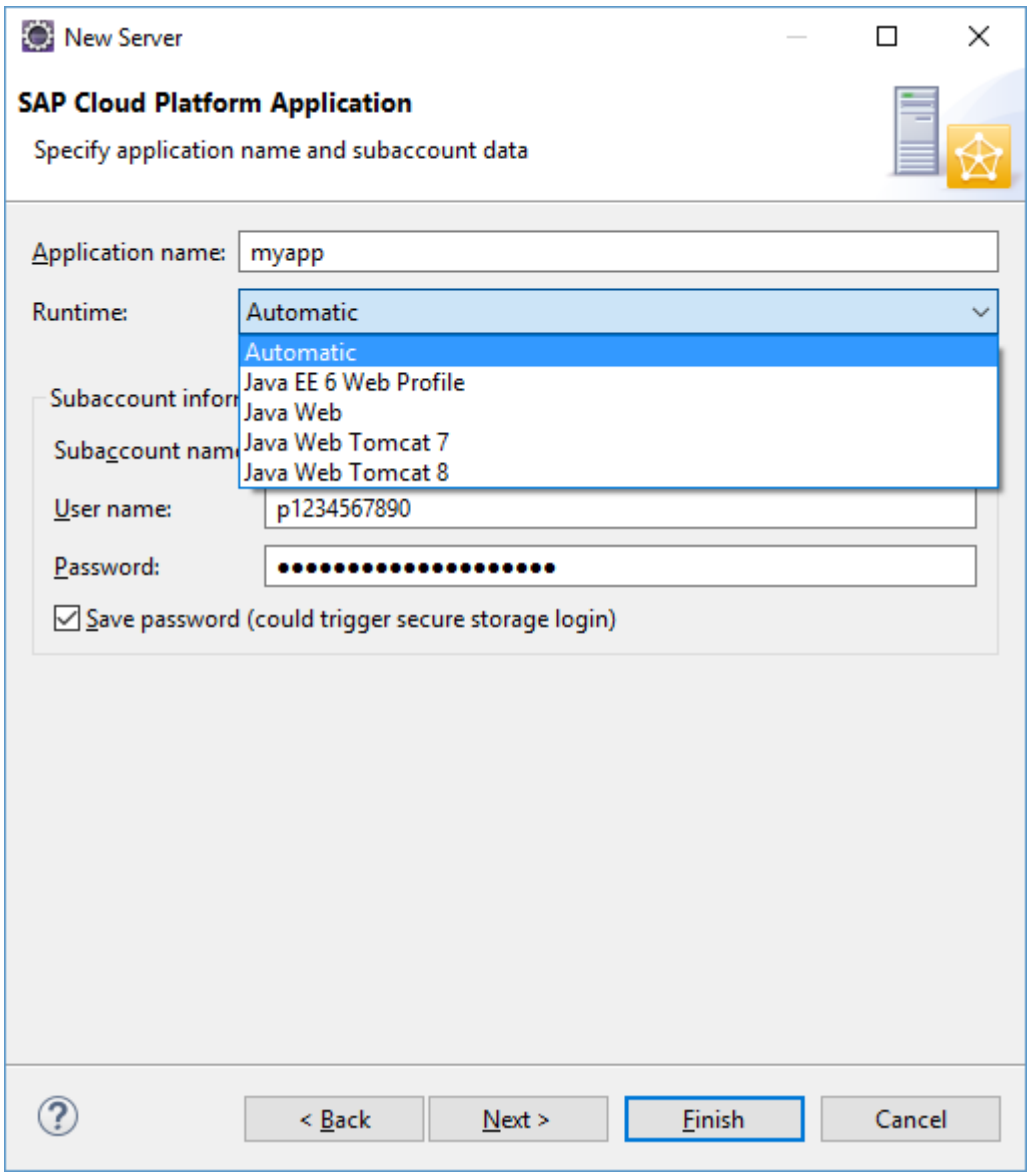
Procedure

1. In the Eclipse IDE, switch to the *Servers* view.
2. Open the context menu define a server with the following settings:
3. Select the server type *SAP* > *SAP Cloud Platform*.
4. Use the host depending on your global account type, then choose *Next*. For more information, see Regions.
5. Specify a unique and identifiable application name.

   Only lowercase Latin letters and digits are allowed.
6. Select a runtime.

   If you leave the default `Automatic` option, the server loads the target runtime of your application.
7. Enter your subaccount name, e-mail or user name, and password, then choose *Next*.
   - If you have previously entered a subaccount and user name for your host, you can select these names from lists.
   - Previously entered hosts also appear in a dropdown list.
   - Select *Save password* to remember and store the password for a given user.

> ⚠ Caution
>
> Do not select your application on the *Add and Remove* screen. Adding an application automatically starts it with the effect that it will fail because no data source binding exists. You will add an application in a later step.



8. Choose *Finish*.
9. In the *Servers* view, open the context menu for the server you just created and choose *Show In* > *Cockpit*.

   The cockpit opens inside Eclipse.

4.2.8.3.1.8 Configure Applications Using the Cockpit

Use the cockpit to create a default binding for your application.

Procedure

1. In the cockpit, select a subaccount, then choose *SAP HANA / SAP ASE* > *Databases & Schemas* in the navigation area.
2. Select the database you want to create a binding for.
3. Choose *Data Source Bindings*. For more information, see .
4. Define a binding (<DEFAULT>) for the application and select a database ID. Choose *Save*.

   You can use an existing database or create a new one.

4.2.8.3.1.9 Start Applications Using Eclipse

Add the application to the new server and start it to deploy the application on the cloud.

Context

> **i Note**
>
> You cannot deploy multiple applications on the same application process. Deployment of a second application on the same application process overwrites any previous deployments. To deploy several applications, deploy each of them on a separate application process.

Procedure

1. On the *Servers* view in Eclipse, open the context menu for the server choose *Add and Remove...* > *<application name>*.
2. To add the application to the server, add the application to the panel on the right side.
3. Choose *Finish*.
4. Start the server.

   This deploys the application and starts it on the SAP BTP.
5. Access the application by clicking the application URL on the application overview page in the cockpit.

   You should see the same output as when the application was tested on the local server.

4.2.8.4 Testing on the Local Runtime

Test an application in the Neo environment that uses the default data source and runs locally on Apache Derby on the local runtime.

If an application uses the default data source and runs locally on Apache Derby, provided as standard for local development, you can test it on the local runtime without any further configuration. To use explicitly named data sources or a different database, you'll need to configure the `connection.properties` file appropriately.

4.2.8.4.1 Configure Data Sources As Connection Properties

To test an application on the local server, define any data sources the application uses as connection properties for the local database. You don't need to do this if the application uses the default data source.

Prerequisites

Start the local server at least once (with or without the application) to create the relevant folder.

Procedure

1. In the *Project Explorer* view, open the folder `Servers/SAP Cloud Platform local runtime/config_master/connection_data` and select *connection.properties*.
2. From the context menu, choose *Open With* > *Properties File Editor*.
3. Add the connection parameter `com.sap.cloud.persistence.dsname` to the block of connection parameters for the local database you are using, as shown in the example below:

   ```
   com.sap.cloud.persistence.dsname=jdbc/datasource1
   javax.persistence.jdbc.driver=org.apache.derby.jdbc.EmbeddedDriver
   javax.persistence.jdbc.url=jdbc:derby:memory:DemoDB;create=true
   javax.persistence.jdbc.user=demo
   javax.persistence.jdbc.password=demo
   eclipselink.target-database=Derby
   ```

   If the application has been bound to the data source based on an explicitly named data source instead of using the default data source, ensure the following:
   - Provide a data source name in the connection properties that matches the name used in the data source binding definition.
   - Add prefixes before each property in a property group for each data source binding you define. If an application is bound only to the default data source, this configuration is considered the default no matter which name you specified in the connection properties. The application can address the data source by any name.
4. Repeat this step for all data sources that the application uses.

5. For the Java EE 6 Web Profile runtime, add the connection parameter `com.sap.cloud.persistence.dsname` twice, once for the managed data source and once for the unmanaged data source, with the names given below. Add each entry in its own block of connection properties:

```
com.sap.cloud.persistence.dsname=jdbc/defaultManagedDataSource
com.sap.cloud.persistence.dsname=jdbc/defaultUnmanagedDataSource
```

6. To indicate that a block of parameters belong together, add a prefix to the parameters, as shown in the example below. The prefix is freely definable; the dot isn't required:

```
1.com.sap.cloud.persistence.dsname=jdbc/datasource1
1.javax.persistence.jdbc.driver=org.apache.derby.jdbc.EmbeddedDriver
1.javax.persistence.jdbc.url=jdbc:derby:memory:DemoDB;create=true
1.javax.persistence.jdbc.user=demo
1.javax.persistence.jdbc.password=demo
1.eclipselink.target-database=Derby
2.com.sap.cloud.persistence.dsname=jdbc/defaultManagedDataSource
2.javax.persistence.jdbc.driver=org.apache.derby.jdbc.EmbeddedDriver
2.javax.persistence.jdbc.url=jdbc:derby:memory:DemoDB;create=true
2.javax.persistence.jdbc.user=demo
2.javax.persistence.jdbc.password=demo
2.eclipselink.target-database=Derby
3.com.sap.cloud.persistence.dsname=jdbc/defaultUnmanagedDataSource
3.javax.persistence.jdbc.driver=org.apache.derby.jdbc.EmbeddedDriver
3.javax.persistence.jdbc.url=jdbc:derby:memory:DemoDB;create=true
3.javax.persistence.jdbc.user=demo
3.javax.persistence.jdbc.password=demo
3.eclipselink.target-database=Derby
```

7. Save the file.
8. Start or restart the server so that the new properties are read.

4.2.8.5 Investigating Performance Issues Using the SQL Trace

Identify inefficient SQL statements in your applications in the Neo environment and investigate performance issues.

Context

The SQL trace provides a log of selected SQL statements with details about when a statement was executed and its duration, allowing you to identify inefficient SQL statements in your applications and investigate performance issues. SQL trace records are integrated in the standard trace log files written at runtime.

By default, the SQL trace is disabled. Generally, you enable it when you require SQL trace information for a particular application and disable it again once you have completed your investigation. It is not intended for general performance monitoring.

You can use the cockpit to enable the SQL trace by setting the log level of the logger `com.sap.core.persistence.sql.trace` to the log level *DEBUG* in the application's log configuration. Once you've changed this setting, you can view trace information in the log files.

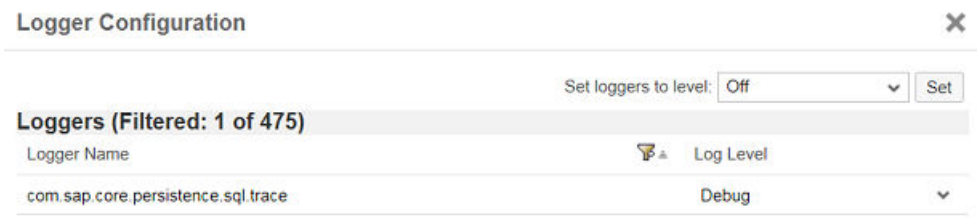
Enable the SQL Trace

Procedure

1. In the SAP BTP cockpit, navigate to a subaccount. For more information, see Navigate in the Cockpit.
2. In the navigation are, choose *Applications* > *Java Applications*.
3. Click the relevant application to go to the dashboard.
4. In the navigation area, choose *Monitoring* > *Logging*.
5. Choose *Configure Loggers*.
   In the *Logger Configuration* dialog box, all loggers used since the application was started are listed with the log levels that are currently applicable.

   > **i Note**
   >
   > You can set log levels only when an application is running. Loggers are not listed if the relevant application code has not been executed.

6. Enter **`com.sap.core.persistence.sql.trace`** in the *Filter* field.
7. In the row containing the `com.sap.core.persistence.sql.trace` logger, select the log level **`Debug`**:



   The new log setting takes effect immediately. Log settings are saved permanently and do not revert to their initial values when an application is restarted.

View the SQL Trace Details

Procedure

1. See the application's trace logs, which contain the SQL trace records, either in the *Most Recent Logging* panel on the application dashboard or on the *Logging* page by navigating to *Monitoring* > *Logging* in the navigation area.
2. To display the contents of a particular log file, choose (*Show*). You can also download the file by choosing (*Download*).

In the log file, you can identify the SQL trace information by the logger name `com.sap.core.persistence.sql.trace`.The entries written by the logger include the following details:

- Date and time written
- System time in nanoseconds
- The name of the interface and method that produced the log entry, for example, `java.sql.Connection.prepareStatement (sql)`
- The status of the method call (`begin` and `end`)
- The database connection ID, for example, `conn=[3d194ab9]`
- The text of the SQL statement, for example, `"INSERT INTO T_PERSONS (ID, FIRSTNAME, LASTNAME) VALUES (?, ?, ?)"`. For security reasons, parameter values are not shown.
- Duration of the request (in milliseconds with microsecond precision), for example, `Request duration = 2,770.743µs`

Example

The SQL-specific information from the default trace is shown below in plain text format:

```
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:03 UTC 2014 -
4913676098596447 - javax.sql.DataSource.getConnection() - begin|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:03 UTC 2014 -
4913676344945627 - javax.sql.DataSource.getConnection() - end - conn=[3d194ab9]
- Request duration = 227,773.804µs - Network traffic = 1462 bytes|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:03 UTC 2014 -
4913676677332569 - java.sql.Connection.prepareStatement(sql) - begin -
conn=[3d194ab9]|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:03 UTC 2014 -
4913676723144228 - java.sql.Connection.prepareStatement(sql) - end -
conn=[3d194ab9] - Request duration = 2,850.647µs - Network traffic = 384 bytes|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:03 UTC 2014 -
4913676723530647 - java.sql.PreparedStatement.executeUpdate() - begin -
conn=[3d194ab9] - SQL="CREATE TABLE T_PERSONS (ID VARCHAR(255) PRIMARY KEY NOT
NULL, FIRSTNAME VARCHAR (255),LASTNAME VARCHAR (255))"|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:03 UTC 2014 -
4913676736488953 - java.sql.PreparedStatement.executeUpdate() - end -
conn=[3d194ab9] - Request duration = 12,760.375µs - Network traffic = 272 bytes|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:03 UTC 2014 -
4913676767170228 - java.sql.Connection.prepareStatement(sql) - begin -
conn=[3d194ab9]|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:03 UTC 2014 -
4913676781082955 - java.sql.Connection.prepareStatement(sql) - end -
conn=[3d194ab9] - Request duration = 12,987.704µs - Network traffic = 464 bytes|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:03 UTC 2014 -
4913676781376595 - java.sql.PreparedStatement.executeQuery() - begin -
conn=[3d194ab9] - SQL="SELECT ID, FIRSTNAME, LASTNAME FROM T_PERSONS"|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:03 UTC 2014 -
4913676786626464 - java.sql.PreparedStatement.executeQuery() - end -
conn=[3d194ab9] - Request duration = 5,118.69µs - Network traffic = 264 bytes|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:09 UTC 2014 -
4913682088475475 - java.sql.Connection.prepareStatement(sql) - begin -
conn=[3d194ab9]|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:09 UTC 2014 -
4913682093620026 - java.sql.Connection.prepareStatement(sql) - end -
conn=[3d194ab9] - Request duration = 4,676.661µs - Network traffic = 392 bytes|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:09 UTC 2014 -
4913682094713377 - java.sql.PreparedStatement.executeUpdate() - begin -
conn=[3d194ab9] - SQL="INSERT INTO T_PERSONS (ID, FIRSTNAME, LASTNAME) VALUES
(?, ?, ?)"|
```

```
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:09 UTC 2014 -
4913682097611865 - java.sql.PreparedStatement.executeUpdate() - end -
conn=[3d194ab9] - Request duration = 2,770.743µs - Network traffic = 336 bytes|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:09 UTC 2014 -
4913682099273612 - java.sql.Connection.prepareStatement(sql) - begin -
conn=[3d194ab9]|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:09 UTC 2014 -
4913682100587082 - java.sql.Connection.prepareStatement(sql) - end -
conn=[3d194ab9] - Request duration = 1,097.86µs - Network traffic = 464 bytes|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:09 UTC 2014 -
4913682100784872 - java.sql.PreparedStatement.executeQuery() - begin -
conn=[3d194ab9] - SQL="SELECT ID, FIRSTNAME, LASTNAME FROM T_PERSONS"|
#DEBUG#com.sap.core.persistence.sql.trace#Thu Apr 10 13:25:09 UTC 2014 -
4913682104569784 - java.sql.PreparedStatement.executeQuery() - end -
conn=[3d194ab9] - Request duration = 3,626.846µs - Network traffic = 308 bytes|
```

Related Information

Alternative Tools for Enabling the SQL Trace [page 982]
Using Logs in the Cockpit

4.2.8.5.1 Alternative Tools for Enabling the SQL Trace

In addition to using the cockpit, you can also enable the SQL trace from the Eclipse IDE, and using the console client. Whichever tool you use, you need to set the log level of the logger `com.sap.core.persistence.sql.trace` to the log level *DEBUG*.

Console Client

You can use the console client to set the log level as a logging property for one or more loggers. To do so, use the command `neo set-log-level` with the log parameters `logger <logger_name>` and `level <log_level>`.

See Using Logs in the Console Client

Related Information

Deploy on the Cloud with the Console Client [page 891]

4.3 Developing SAP HANA in the Neo Environment

With SAP BTP, you can use the SAP HANA development tools to create comprehensive analytical models and build applications with SAP HANA's programmatic interfaces and integrated development environment.

Benefits and advantages

- Automatic backups
- Creation of SAP HANA schemas and repository packages. Your SAP HANA instances and XS applications are visualized in the cockpit.
- Eclipse-based tools for connecting to your SAP HANA instances on SAP BTP
- Eclipse-based tools for data modeling

Appropriate for

- Implementing highly intensive calculation scenarios
- Building analytic models
- Implementing big data scenarios
- Implementing Internet of Things scenarios
- Building XS applications
- Using SAP HANA embedded search capabilities
- Leveraging SAP HANA functional libraries like SAP HANA Business Function Library (BFL) and the SAP HANA Predictive Analytics Library (PAL)
- Developing hybrid applications (native HANA, Java, HTML5, mobile)

Not appropriate for

- Applications requiring OS-level access

Related Information

SAP HANA: Getting Started [page 983]
Using an SAP HANA XS Database System [page 999]

4.3.1 SAP HANA: Getting Started

Set up your SAP HANA development environment and run your first application in the cloud.

1. Getting a Global Account
   You first need to sign up for an SAP BTP account.
2. Set Up
   Download Eclipse IDE for Java EE Developers, and set up SAP HANA Tools.
3. Create
   Create a simple SAP HANA XS application using SAP HANA Web-based Development Workbench and run it in the cloud.

You can also create an SAP HANA XS application using SAP HANA Studio [page 989].

4. Monitor
   Monitor SAP HANA XS applications.

Add Features

Use calculation views and visualize the data with SAPUI5. See: 8 Easy Steps to Develop an XS application on the SAP BTP

Enable SHINE

Enable the demo application SAP HANA Interactive Education (SHINE) [page 996] and learn how to build native SAP HANA applications.

4.3.1.1 Install SAP HANA Tools for Eclipse

Download and set up the necessary tools.

Prerequisites

- You have downloaded and installed a version of Eclipse IDE.

  > → Recommendation
  >
  > Use the latest version.

- You have configured your proxy settings (in case you work behind a proxy or a firewall). For more information, see → **step 3**.

Procedure

1. Open the Eclipse IDE.
2. In the main menu, choose *Help* > *Install New Software*.
3. In the *Work with* field, enter the following URL: `https://tools.hana.ondemand.com/latest/`
4. Select *SAP HANA Tools* (the whole feature group).

> **Note**
>
> In case you need to develop with SAPUI5, install also *Tools* > *UI development toolkit for HTML5 (Developer Edition)*.

5. Choose *Next*.
6. On the next wizard page, you get an overview of the features to be installed. Choose *Next*.
7. Confirm the license agreements.
8. Choose *Finish* to start the installation.
9. After the successful installation, you will be prompted to restart your Eclipse IDE.

Next Steps

Creating an SAP HANA XS Hello World Application Using SAP HANA Web-based Development Workbench [page 985]

Creating an SAP HANA XS Hello World Application Using SAP HANA Studio [page 989]

4.3.1.2 Creating an SAP HANA XS Hello World Application Using SAP HANA Web-based Development Workbench

Create and test a simple SAP HANA XS application that displays the "Hello World" message using the SAP HANA Web-Based Development Workbench.

Prerequisites in Enterprise Accounts

- Install an SAP HANA tenant database system (MDC). See Install Database Systems.
- You are assigned the Administrator role for the subaccount.

4.3.1.2.1 Create a Database

In your subaccount in the SAP BTP cockpit, you create a database on an SAP HANA tenant database system.

Procedure

1. In the SAP BTP cockpit, navigate to a subaccount. For more information, see Navigate in the Cockpit [page 1259].
2. In the navigation area, choose *SAP HANA / SAP ASE* > *Databases & Schemas*.
3. From the *Databases & Schemas* page, choose *New*.
4. Enter the required data:

Property	Value
Database ID	Example: **`tutorial`**
Database System	An SAP HANA tenant database system. *Example:* `mdc1 (HANAMDC)` **i Note** `mdc1` corresponds to the database system on which you create the database.
SYSTEM User Password	The password for the SYSTEM user of the database. **i Note** The SYSTEM user is a preconfigured database super-user with irrevocable system privileges, such as the ability to create other database users, access system tables, and so on. A database-specific SYSTEM user exists in every database of a tenant database system.

5. Choose *Create*.
6. The *Events* page shows the progress of the database creation. Wait until the tenant database is in state *Started*.
7. (Optional) To view the details of the new database, choose *Overview* in the navigation area and select the database in the list. Verify that the status *STARTED* is displayed.

4.3.1.2.2 Create a Database User and Add the Required Roles

Create a new database user in the SAP HANA cockpit and assign the user the required permissions for working with the SAP HANA Web-based Development Workbench.

Context

You've specified a password for the SYSTEM user when you created an SAP HANA tenant database. You now use the SYSTEM user to log on to SAP HANA cockpit and create your own database administration user.

> ⚠ Caution
>
> You should not use the SYSTEM user for day-to-day activities. Instead, use this user to create dedicated database users for administrative tasks and to assign privileges to these users.

Procedure

1. Log on to the SAP HANA cockpit:
   a. In the navigation area of the SAP BTP cockpit, choose *SAP HANA / SAP ASE* > *Databases & Schemas*.
   b. Select the relevant database.
   c. In the database overview, open the SAP HANA cockpit link under *Administration Tools*.
   d. In the SAP HANA cockpit, enter **`SYSTEM`** as the user, and its password.

      A message appears, telling you that you do not have the required roles.
   e. Choose *OK*.

      You receive a confirmation that the required roles are assigned to you automatically.
   f. Choose *Continue*.
2. Choose *Manage Roles and Users*.
3. Expand the *Security* node.
4. Open the context menu for the *Users* node and choose *New User*.
5. On the *User* tab, provide a name for the new user.

   The user name will always appear in uppercase letters.
6. In the *Authentication* section, make sure the *Password* checkbox is selected and enter a password.

   The password must start with a letter and only contain uppercase and lowercase letters ('a' – 'z', 'A' – 'Z'), and numbers ('0' – '9').
7. Save your changes.
8. In the *Granted Roles* section, choose + (Add Role).

9. Type **ide** in the search field and select all roles in the result list.

   The roles are added on the *Granted Roles* tab.
10. Save your changes.
11. To assign the `CONTENT_ADMIN` role to the user, repeat the steps in the *Granted Roles* section, searching for `CONTENT_ADMIN`.
12. Before you continue to work with the SAP HANA Web-based Development Workbench, you log out first and log on again with your new database user.

    > ⚠ Caution
    >
    > At this point, you are still logged on with the SYSTEM user. You can only use your new database user to work with SAP HANA Web-based Development Workbench by logging out from SAP HANA cockpit first. Otherwise, you would automatically log in to the SAP HANA Web-based Development Workbench with the SYSTEM user instead of your new database user. Therefore, choose the *Logout* button before you continue to work with the SAP HANA Web-based Development Workbench, where you need to log on again with the new database user.
13. (Optional) Disable the Password Lifetime Handling for a New Technical SAP HANA Database User.

    This step is not necessary to complete this tutorial, but you shouldn't forget to disable the password lifetime handling in productive scenarios.

4.3.1.2.3 Create and Deploy an SAP HANA XS Application

Create an SAP HANA XS Hello World program using the SAP HANA Web-based Development Workbench.

Procedure

1. In the navigation area of the SAP BTP cockpit, choose *SAP HANA / SAP ASE* > *Databases & Schemas*.
2. Select the relevant database.
3. In the database overview, open the *SAP HANA Web-based Development Workbench* link under *Development Tools*.
4. Log on to the SAP HANA Web-based Development Workbench with your new database user and password.
5. Select the *Editor*.

   > → Tip
   >
   > The editor header shows details for your user and database. Hover over the entry for the SID to view the details.
6. To create a new package, choose *New* > *Package* from the context menu of the *Content* folder.
7. Enter a package name.

   The package appears under the *Content* folder node.

8. From the context menu for the new package node, choose *Create Application*.
9. Select *HANA XS Hello World* as template and choose *Create*.
10. To deploy the program, select the `logic.xsjs` file from the new package and choose *Run*.

    The program is deployed and appears in the browser: `Hello World from User <Your User>.`

4.3.1.3 Creating an SAP HANA XS Hello World Application Using SAP HANA Studio

Create and test a simple SAP HANA XS application that displays the "Hello World" message.

Prerequisites

Make sure the database you want to use is deployed in your account before you begin with this tutorial. For more information, see Getting Started. You can create SAP HANA XS applications using one of the following database systems:

- SAP HANA single-container database system (XS)
- SAP HANA tenant database database system (MDC)

You also need to install the tools as described in Install SAP HANA Tools for Eclipse [page 984] to follow the steps described in this tutorial.

Context

In this tutorial, you complete the following steps:

1. Create a Database User [page 989]
2. Add a Repository Workspace [page 991]
3. Add an XS Application Project [page 992]
4. Write Server-Side JavaScript [page 993]
5. Test Your Application [page 994]
6. Retrieve Data from SAP HANA [page 994]
7. Test Your Application Again [page 995]

4.3.1.3.1 Create a Database User

Context

You will perform all subsequent activities with this new user.

Procedure

1. Log on to the SAP BTP cockpit and select an account.
2. Choose *SAP HANA / SAP ASE* > *Databases & Schemas* in the navigation area.

   All databases available in the selected account are listed with their ID, type, version, and related database system.

   > → Tip
   >
   > To view the details of a database, for example, its state and the number of existing bindings, select a database in the list and click the link on its name. On the overview of the database, you can perform further actions, for example, delete the database.

3. Depending on the database you are using, choose one of the following options:

If you want to create your application using...	Do the following:
An SAP HANA XS database	Follow the steps described in Create a Database Administrator User [page 1003].
An SAP HANA tenant database	1. Select the relevant SAP HANA tenant database in the list. 2. In the overview that is shown in the lower part of the screen, open the *SAP HANA cockpit* link under *Administration Tools*. 3. In the *Enter Username* field, enter **`SYSTEM`**, then enter the password you determined for the SYSTEM user in the *Enter Password* field. A message is displayed to inform you that at that point, you lack the roles that you need to open the SAP HANA cockpit. 4. To confirm the message, choose *OK*. You receive a confirmation that the required roles are assigned to you automatically. 5. Choose *Continue*. You are now logged on to the SAP HANA cockpit. 6. Choose *Manage Roles and Users*. 7. To create database users and assign them the required roles, expand the *Security* node. 8. Open the context menu for the *Users* node and choose *New User*. 9. On the *User* tab, provide a name for the new user. The user name always appears in upper case letters. 10. In the *Authentication* section, make sure the *Password* checkbox is selected and enter a password. > ⓘ Note > The password must start with a letter and only contain uppercase and lowercase letters ('a' - 'z', 'A' - 'Z'), and numbers ('0' - '9'). 11. To create the user, choose *Save*. The new database user is displayed as a new node under the *Users* node. 12. To assign your user the roles with the required permissions for working with SAP HANA Web-based Development Workbench, go to the *Granted Roles* section and choose the *+ (Add Role)* button. 13. Type **`xs.ide`** in the search field and select all roles in the result list. 14. Choose *Ok*. The roles are added on the *Granted Roles* tab.

If you want to create your application using...	Do the following:
	15. Repeat the last two steps to assign the **CONTENT_ADMIN** role to the user. **i Note** For more information on the CONTENT_ADMIN role, see Predefined Database Roles. 16. Save your changes. **⚠ Caution** At this point, you are still logged on with the SYSTEM user. You can only use your new database user to work with SAP HANA Web-based Development Workbench by logging out from SAP HANA Cockpit first. Otherwise, you would automatically log in to the SAP HANA Web-based Development Workbench with the SYSTEM user instead of your new database user. Therefore, choose the *Logout* button before you continue to work with the SAP HANA Web-based Development Workbench, where you need to log on again with the new database user.

4.3.1.3.2 Add a Repository Workspace

Context

After you add the SAP HANA system hosting the repository that stores your application-development files, you must specify a repository workspace, which is the location in your file system where you save and work on the development files.

Procedure

1. In the Eclipse IDE, go to *Window* > *Perspective* > *Open Perspective* > *Other*.
2. Select *SAP HANA Development* and choose *OK*.
3. In the *Repositories* view, choose *File* > *New* > *Repository Workspace*.
   - SAP HANA system:
     Choose the same system you just added for this tutorial.
   - Workspace Name:
     If a default workspace exists, uncheck the *Default workspace* option and enter a workspace name.
     A folder with the name you type is created below the *Workspace Root*.
   - Workspace root:
     The *Workspace Root* is a folder that contains the workspace you create in this step. The *Workspace Root* can be anywhere on your local file system.
4. Click *Finish*.

Results

In the *Repositories* view, you see your workspace, which enables you to browse the repository of the system tied to this workspace. The repository packages are displayed as folders.

At the same time, a folder will be added to your file system to hold all your development files.

4.3.1.3.3 Add an XS Application Project

Context

After you set up a development environment for the chosen SAP HANA system, you can add a project to contain all the development objects you want to create as part of the application-development process. There are a variety of project types for different types of development objects. Generally, a project type ensures that only the necessary libraries are imported to enable you to work with development objects that are specific to a project type. In this tutorial, you create an XS Project.

Procedure

1. In the *SAP HANA Development* perspective in the Eclipse IDE, choose *File* > *New* > *XS Project*.
2. Make sure the *Share project in SAP repository* option is selected and enter a project name.
3. Choose *Next*.
4. Select the repository workspace you created in the previous step and choose *Next*.
5. Choose *Finish* without doing any further changes.

Results

The *Project Explorer* view in the *SAP HANA Development* perspective in Eclipse displays the new project. The system information in brackets to the right of the project node name in the *Project Explorer* view indicates that the project has been shared; shared projects are regularly synchronized with the Repository hosted on the SAP HANA system you are connected to.

4.3.1.3.4 Write Server-Side JavaScript

Context

SAP HANA Extended Application Services (SAP HANA XS) supports server-side application programming in JavaScript. In this step, you add some simple JavaScript code that generates a page which displays the words `Hello, World!`

Procedure

1. In the *Project Explorer* view in the *SAP HANA Development* perspective in Eclipse, right-click your XS project, and choose *New* > *Other*.
2. In the *Select a wizard* dialog, choose *SAP HANA* > *Application Development* > *XS JavaScript File*.
3. In the *New XS JavaScript File* dialog, enter **`MyFirstSourceFile.xsjs`** as *File name* and choose *Next*.
4. Choose *Finish*.
5. In the `MyFirstSourceFile.xsjs` file, enter the following code and save the file:

   ```
   $.response.contentType = "text/html";
   $.response.setBody( "Hello, World !");
   ```

   > **i Note**
   >
   > By default, saving the file automatically commits the saved version of the file to the repository.

   The example code shows how to use the SAP HANA XS JavaScript API's `response` object to write HTML. By typing `$.` you have access to the API's objects.
6. Check that the application descriptor files (`xs.app` and `xs.access`) are present in the root package of your new XS JavaScript application.

   The application descriptors are mandatory and describe the framework in which an SAP HANA XS application runs. The `.xsapp` file indicates the root point in the package hierarchy where content is to be served to client requests; the `.xsaccess` file defines who has access to the exposed content and how.

   > **i Note**
   >
   > By default, the project-creation Wizard creates the application descriptors automatically. If they aren't present, you see a 404 error message in the Web Browser when you call the XS JavaScript service. In this case, you need to create the application descriptors manually. See the SAP HANA Developer Guide for SAP HANA Studio for instructions.
7. Open the context menu for the new files (or the folder/package containing the files) and select *Team* > *Activate All*. The activate operation publishes your work and creates the corresponding catalog objects; you can now test it.

4.3.1.3.5 Test Your Application

Context

Check if your application is working and if the `Hello, World!` message is displayed.

Procedure

In the *SAP HANA Development* perspective in the Eclipse IDE, open the context menu of the `MyFirstSourceFile.xsjs` file and choose *Run As* > *1 XS Service*.

> **i Note**
>
> You might need to enter the credentials of the database user you created in this tutorial again.

> **i Note**
>
> If you have used an SAP HANA XS database for creating your SAP HANA XS application, you can also launch your application from the SAP BTP cockpit by choosing the application URL after navigating to *Applications* > *HANA XS Applications*. For more information, see Launch SAP HANA XS Applications [page 998].

Results

The following text should be displayed:

```
Hello, World !
```

4.3.1.3.6 Retrieve Data from SAP HANA

Context

To extract data from the database, you use your JavaScript code to open a connection to the database and then prepare and run an SQL statement. The results are added to the `Hello, World!` response. You use the following SQL statement to extract data from the database:

```
select * from DUMMY
```

The SQL statement returns one row with one field called *DUMMY*, whose value is *X*.

Procedure

1. In the *Project Explorer* view in the *SAP HANA Development* perspective in Eclipse, open the `MyFirstSourceFile.xsjs` file in the embedded JavaScript editor.
2. In the `MyFirstSourceFile.xsjs` file, replace your existing code with the following code:

```
$.response.contentType = "text/html";
var output = "Hello, World !";
var conn = $.db.getConnection();
var pstmt = conn.prepareStatement( "select * from DUMMY" );
var rs = pstmt.executeQuery();
if (!rs.next()) {
    $.response.setBody( "Failed to retrieve data" );
    $.response.status =  $.net.http.INTERNAL_SERVER_ERROR;
} else {
    output = output + "This is the response from my SQL: " + rs.getString(1);
}
rs.close();
pstmt.close();
conn.close();
$.response.setBody(output);
```

3. Save the file `MyFirstSourceFile.xsjs`.
4. Open the context menu of the `MyFirstSourceFile.xsjs` file and choose *Team* > *Activate All*.

4.3.1.3.7 Test Your Application Again

Context

Check if your application is retrieving data from your SAP HANA database.

Procedure

In the *SAP HANA Development* perspective in the Eclipse IDE, open the context menu of the `MyFirstSourceFile.xsjs` file and choose *Run as* > *XS Service*.

> **i Note**
>
> If you have used an SAP HANA XS database for creating your SAP HANA XS application, you can also launch your application from the SAP BTP cockpit by choosing the application URL after navigating to *Applications* > *HANA XS Applications*. For more information, see Launch SAP HANA XS Applications [page 998].

Results

The following text should be displayed:

```
Hello, World!This is the reponse from my SQL: X
```

4.3.1.4 Enable SAP HANA Interactive Education (SHINE)

You can enable the SAP HANA Interactive Education (SHINE) demo application for a new or existing SAP HANA tenant database.

Context

> **!** Restriction
>
> SHINE is not available on the China (Shanghai) region.

SAP HANA Interactive Education (SHINE) demonstrates how to build native SAP HANA applications. The demo application comes with sample data and design-time developer objects for the application's database tables, data views, stored procedures, OData, and user interface. For more information, see the SAP HANA Interactive Education (SHINE) documentation.

By default, SHINE is available for all SAP HANA tenant databases in trial accounts in the Neo environment.

Procedure

1. Log in to the SAP BTP cockpit navigate to a subaccount. For more information, see Navigate in the Cockpit.
2. In the navigation area, choose *SAP HANA / SAP ASE* > *Databases & Schemas*.
3. To enable SHINE for an SAP HANA tenant database, you must first create a SHINE user. If you are enabling SHINE for a new SAP HANA tenant database, a SHINE user can be automatically created during the database creation. If you are enabling SHINE for an existing SAP HANA tenant database, you must manually create the SHINE user.

To...	Do the following:
Enable SHINE for a new SAP HANA tenant database	1. Follow the steps described in Create SAP HANA Tenant Databases. 2. From the list of all databases and schemas, choose the SAP HANA tenant database you just created. 3. In the overview in the lower part of the screen, choose the *SAP HANA Interactive Education (SHINE)* link under *Education Tools*.

To...	Do the following:
Enable SHINE for an existing SAP HANA tenant database	1. From the list of all databases and schemas, choose the SAP HANA tenant database for which you want to enable SHINE. 2. In the overview in the lower part of the screen, open the *SAP HANA Cockpit* link under *Administration Tools*. 3. In the *Enter Username* field, enter **`SYSTEM`**, then enter the password you determined for the SYSTEM user. The first time you log in to the *SAP HANA Cockpit*, you are informed that you don't have the roles that you need to open the SAP BTP cockpit. 4. Choose *OK*. The required roles are assigned to you automatically. 5. Choose *Continue*. You are now logged in to the *SAP HANA Cockpit*. 6. Choose *Manage Roles and Users*. 7. To create database users and assign them the required roles, expand the *Security* node. 8. Open the context menu for the *Users* node and choose *New User*. 9. On the *User* tab, provide a name for the new SHINE user. **i Note** The user name can contain only uppercase and lowercase letters ('a' - 'z', 'A' - 'Z'), numbers ('0' - '9'), and underscores ('_'). The user name always appears in uppercase letters. 10. In the *Authentication* section, make sure the *Password* option is selected and enter a password. **i Note** The password must contain at least one uppercase and one lowercase letter ('a' - 'z', 'A' - 'Z') and one number ('0' - '9'). It can also contain special characters (except ", ' and \). 11. Choose *Save*. The new database user appears as a new node under the *Users* node. 12. In the *Granted Roles* section, select *+ (Add Role)*. 13. Enter **`democontent`** in the search field and select all roles in the result list. 14. Choose *Ok*. The roles are added to the *Granted Roles* tab. 15. Save your changes. 16. On the *Overview* page of your SAP HANA tenant database in the *cockpit*, choose the *SAP HANA Interactive Education (SHINE)* link under *Education Tools* in the lower part of the screen.

A login screen for the SHINE demo application is shown in a new browser window.

4. Enter the credentials of the SHINE user you created and choose *Login*.

i Note

The first time you log in to the SHINE demo application, you are prompted to change your initial password.

Results

You see the SHINE demo application for your SAP HANA tenant database. Consult the SAP HANA Interactive Education (SHINE) documentation for detailed information about using the application.

4.3.2 Launch SAP HANA XS Applications

You can open your SAP HANA XS applications in a Web browser directly from the cockpit.

Context

> **i Note**
>
> This feature is only available for SAP HANA XS applications in single container SAP HANA systems. For SAP HANA XS applications in SAP HANA tenant database systems, use SAP Web IDE or SAP HANA cockpit to manage your applications.

Procedure

1. In the SAP BTP cockpit, navigate to a subaccount. For more information, see Navigate in the Cockpit.
2. Choose *Applications* > *HANA XS Applications*.
3. In the *HANA XS Applications* table, click the application URL link to launch the application.

> **i Note**
>
> If an HTTP status 404 (not found) error is shown, bear in mind that the cockpit displays only the root of an application's URL path. This means that you might have to either:
>
> - Add the application name to the URL address in the browser, for example, `hello.xsjs`.
> - Use an `index.html` file, which is the default setting for the file displayed when the package is accessed without specifying a file name in the URL.
> - Override the above default setting by specifying the *default_file* keyword in the `.xsaccess` file, for example:
>
> ```
> {
> "exposed" : true,
> "default_file": "hello.xsjs"
> }
> ```

4.3.3 Using an SAP HANA XS Database System

Use SAP HANA single-container database systems designed for developing with SAP HANA in a productive environment.

SAP HANA XS Database Systems in Enterprise Accounts

Prerequisites

An SAP HANA XS database system is deployed in a subaccount in your enterprise account. For more information, see Install Database Systems.

Performance/Scalability Recommendation

Before going live with an application for which a significant number of users and/or significant load is expected, you should do a performance load test. This is best practice in the industry and we strongly recommend it for HANA XS applications.

Administrative Database Users

SAP BTP creates four users that it requires to manage the database: *SYSTEM*, *BKPMON*, *CERTADM*, and *PSADBA*. These users are reserved for use by SAP BTP. For more information, see Create a Database Administrator User [page 1003].

> ⚠ Caution
>
> Do not delete or deactivate these users or change their passwords.

Technical Database User

Each SAP HANA XS database system has a technical database user `NEO_<guid>`, which is created automatically when the database system is assigned to a subaccount. A technical database user is not the same as a normal database user and is provided purely as a mechanism for enabling schema access.

> ⚠ Caution
>
> Do not delete or change the technical database user in any way (password, roles, permissions, and so on).

Features

See the overview below for details about available features:

Feature	Description
Database users	Specifics for SAP BTP: • Creating Database Users [page 1003] • Create a Database Administrator User [page 1003]
User management	You have full administration privileges to manage users See: • Managing SAP HANA Users • Setting Up Roles and Authorizations • SAP HANA User and Role Management
Database Administration	See Administration
Web-based tools	• SAP HANA Web-based Development Workbench • SAP HANA Cockpit • SAP HANA XS Administration Tool See: • Supported SAP HANA Web-based Tools [page 1001] • Roles Required for Web-based Tools [page 1006] • Developing Applications in Web-based Environments • SAP HANA XS Administration Tools
Eclipse	• Install SAP HANA Tools for Eclipse [page 984] • Connect to SAP HANA Databases via the Eclipse IDE
Security	Choice of authentication method See: • Configure SAML 2.0 Authentication [page 1015] • SAP HANA XS Administration Tools
Connectivity destinations	• Connectivity for SAP HANA XS (Enterprise Version) [page 217] • Maintaining HTTP Destinations
Monitoring	• Configure Availability Checks for SAP HANA XS Applications from the Cockpit [page 1006] • Configure Availability Checks for SAP HANA XS Applications from the Console Client [page 1007] • View Monitoring Metrics of a Database System [page 1009] • View Monitoring Metrics of an SAP HANA XS Application [page 766]

Feature	Description
Debugging	• SAP HANA Web-based Development Workbench See: Debug with SAP HANA Web-based Development Workbench [page 1014] • Supported by the SAP HANA Tools plug-in for Eclipse as of release 7.4.
Launch SAP HANA XS applications	Launch SAP HANA XS Applications [page 998]
Java development	bind-hana-dbms [page 1368]

Related Information

SAP HANA Administration Guide
SAP HANA Security Guide
SAP HANA Developer Guide for SAP HANA Studio
SAP HANA Developer Guide for SAP HANA Web Workbench

4.3.3.1 Supported SAP HANA Web-based Tools

SAP BTP supports the following Web-based tools: SAP HANA Web-based Development Workbench, SAP HANA Studio, and SAP HANA XS Administration Tool.

Prerequisites

- You have a database user. See Creating Database Users [page 1003].
- Your database user is assigned the roles required for the relevant tool. See Roles Required for Web-based Tools [page 1006].

Context

You can access the SAP HANA Web-based tools using the Cockpit or the tool URLs. The following table summarizes what each supported tool does, and how to acess it.

Supported Web-Based Tools for SAP HANA Development and Administration

Tool	Description	Where to Find It in the Cockpit	Tool URL
SAP HANA Web-based Development Workbench	Includes an all-purpose editor tool that enables you to maintain and run design-time objects in the SAP HANA repository. It does not support modeling activities. For more information, see the Developer Guide for SAP HANA Web Workbench.	*Development Tools* section: *SAP HANA Web-based Development Workbench*	`https://<database instance><subaccount>.< host>/sap/hana/xs/ide/`
SAP HANA Cockpit	Provides you with a single point-of -access to a range of Web-based applications for the online administration of SAP HANA. For more information, see the SAP HANA Administration Guide. **i Note** It is not possible to use the SAP HANA database lifecycle manager (HDBLCM) with the cockpit.	*Administration Tools* section: *SAP HANA Cockpit*	`https://<database instance><subaccount>.<host>/sap/hana/xs/admin/cockpit`
SAP HANA XS Administration Tool	Allows you, for example, to configure security options and HTTP destinations. For more information, see the SAP HANA Administration Guide.	*Administration Tools* section: *SAP HANA XS Administration*	`https://<database instance><subaccount>.<host>/sap/hana/xs/admin/`

→ Remember

When using the tools, log on with your database user (**not** your SAP BTP user). If this is your initial logon, you will be prompted to change your password. You are responsible for choosing a strong password and keeping it secure.

Related Information

Regions

Developing Applications in Web-based Environments
Debug with SAP HANA Web-based Development Workbench [page 1014]

4.3.3.2 Creating Database Users

Use the database user feature in the SAP BTP cockpit to create a database administration user for SAP HANA XS databases, and set up database users in SAP HANA for the members of your development team.

To create database users for SAP HANA XS databases, perform the following steps:

1. Create a Database Administrator User [page 1003].
2. Assign the required development team members to your subaccount. See Add Members to Your Neo Subaccount [page 1299].
3. In the SAP HANA database system, create database users for the members of your subaccount and assign them the required developer roles. See Setting Up Roles and Authorizations and Managing SAP HANA Users.
4. (Optional) Assign Roles Required for the SAP HANA XS Administration Tool [page 1005].

Related Information

Roles Required for Web-based Tools [page 1006]

4.3.3.2.1 Create a Database Administrator User

As an subaccount administrator, you can use the database user feature provided in the cockpit to create your own database user for your SAP HANA database.

Context

SAP BTP creates four users that it requires to manage the database: SYSTEM, BKPMON, CERTADM, and PSADBA. These users are reserved for use by SAP BTP.

> ⚠ Caution
>
> Do not delete or deactivate these users or change their passwords.

Procedure

1. In the SAP BTP cockpit, navigate to a subaccount. For more information, see Navigate in the Cockpit [page 1259].
2. Choose *SAP HANA / SAP ASE* > *Databases and Schemas* in the navigation area.

   You see all databases that are available in the subaccount, along with their details, including the database type, version, memory size, state, and the number of associated databases.
3. Select the relevant SAP HANA XS database.
4. In the *Development Tools* section, click *Database User*.
   A message confirms that you do not yet have a database user.
5. Choose *Create User*.
   Your user and initial password are displayed. Change the initial password when you first log on to an SAP HANA system, for example the SAP HANA Web-based Development Workbench.

   > ### ℹ Note
   >
   > - Your database user is assigned a set of permissions for administering the SAP HANA database system, which includes *HCP_PUBLIC*, and *HCP_SYSTEM*. The *HCP_SYSTEM* role contains, for example, privileges that allow you to create database users and grant additional roles to your own and other database users.
   > - You also require specific roles to use the SAP HANA Web-based tools. For security reasons, only the role that provides access to the SAP HANA Web-based Development Workbench is assigned as default.
6. To log on to the SAP HANA Web-based Development Workbench and change your initial password now (recommended), copy your initial password and then close the dialog box.

   You do not have to change your initial password immediately. You can open the dialog box again later to display both your database user and initial password. Since this poses a potential security risk, however, you are strongly advised to change your password as soon as possible.
7. In the *Development Tools* section, click *SAP HANA Web-based Development Workbench*.
8. On the SAP HANA logon screen, enter your database user and initial password.
9. Change your password when prompted.

   > ### ⚠ Caution
   >
   > You are responsible for choosing a strong password and keeping it secure. If your user is blocked or if you've forgotten the password of your user, another database administration user with `USER_ADMIN` privileges can unlock your user.

Next Steps

- > ### → Tip
  >
  > There may be some roles that you cannot assign to your own database user. In this case, we recommend that you create a second database user (for example, *ROLE_GRANTOR*) and assign it the

> *HCP_SYSTEM* role. Then log onto the SAP HANA system with that user and grant your database user the roles you require.

- In the SAP HANA system, you can now create database users for the members of your subaccount and assign them the required developer roles.
- To be able to use other HANA tools like HANA Cockpit or HANA XS Administration tool, you must assign yourself access to these before you can start using them. See Assign Roles Required for the SAP HANA XS Administration Tool [page 1005]

Related Information

Roles Required for Web-based Tools [page 1006]
Managing SAP HANA Users
Setting Up Roles and Authorizations
Creating an SAP HANA Tenant Database and an SAP HANA XS Application

4.3.3.2.2 Assign Roles Required for the SAP HANA XS Administration Tool

To work with the SAP HANA XS Administration Tool, add the required rules to your database user.

Context

The initial set of roles of your database user also contains the *sap.hana.xs.ide.roles::Developer* role, allowing you to work with the SAP HANA Web-based Development Workbench, but not the SAP HANA XS Administration tool.

To be able to work with the SAP HANA XS Administration Tool `(https://<schema><subaccount>.<host>sap/hana/xs/admin/)`, you require the relevant *sap.hana.xs.admin.roles*. However, these are not included in the initial set of roles. To assign these roles to your database user, perform the following steps:

Procedure

1. Log on in with your user to one of the following tools:
   - Use the Eclipse IDE and connect to your SAP HANA studio. For more information, see Connect to SAP HANA Databases via the Eclipse IDE.
   - Use the SAP HANA Web-based Development Workbench. For more information, see Supported SAP HANA Web-based Tools [page 1001].

2. In the *Security* section, assign the required set of roles to yourself. For more information, see Roles Required for Web-based Tools [page 1006].

4.3.3.2.2.1 Roles Required for Web-based Tools

To use the SAP HANA Web-based tools, you require specific roles.

Role	Description
SAP HANA Web-based Development Workbench	
sap.hana.xs.ide.roles::EditorDeveloper or parent role *sap.hana.xs.ide.roles::Developer*	Use the Editor component of the SAP HANA Web-based Development Workbench.
sap.hana.xs.debugger::Debugger	Debug server-side JavaScript code.
SAP HANA XS Administration Tool	
sap.hana.xs.admin.roles::HTTPDestViewer	View HTTP destinations.
sap.hana.xs.admin.roles::HTTPDestAdministrator	Full access to HTTP destination configurations (display and edit).
sap.hana.xs.admin.roles::TrustStoreViewer	Read-only access to the trust store, which contains the server's root certificate or the certificate of the certification authority that signed the server's certificate.
sap.hana.xs.admin.roles::TrustStoreAdministrator	Full access to the SAP HANA XS application trust store to manage the certificates required to start SAP HANA XS applications.

Related Information

Supported SAP HANA Web-based Tools [page 1001]
SAP HANA Platform 1.0 SPS 12

4.3.3.3 Configure Availability Checks for SAP HANA XS Applications from the Cockpit

In the SAP BTP cockpit, you can configure availability checks for the SAP HANA XS applications running on your productive SAP HANA database system.

Prerequisites

- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

- You have deployed and started an SAP HANA XS application in your subaccount.

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Choose *Applications* > *HANA XS Applications* in the navigation area of the subaccount.
3. Select an application from the list, and in the *Application Details* panel, configure the check.
   - To create a new check, choose *Create Availability Check*.
   - To view a created availability check, choose *Details*.
   - To update a created availability check, choose *Details* > *Edit*.
   - To delete a created availability check, choose *Details* > *Delete*.
4. In the dialog that appears, select or enter the URL you want to monitor and fill in values for warning and critical thresholds if you want them to be different from the current ones. The default values are 50 seconds for a warning threshold and 60 seconds for a critical threshold.
5. Save your configuration.

   When your availability check is created, you can view your application's latest HTTP response code and response time as well as a status icon showing whether your application is up or down. If you want to receive alerts when your application is down, you have to configure alerting.

Related Information

Alerting [page 779]
SAP BTP Cockpit
Availability Checks [page 767]
Configure Availability Checks for SAP HANA XS Applications from the Console Client [page 1007]

4.3.3.4 Configure Availability Checks for SAP HANA XS Applications from the Console Client

In the console client, you can configure an availability check for your SAP HANA XS application and subscribe recipients to receive alert e-mail notifications when it is down or responds slowly. For how to configure alerting, see the Related Information section.

Prerequisites

- You have a productive SAP HANA database on the platform.
  For more information, see Using an SAP HANA XS Database System [page 999].

- You have set up the console client.
  For more information, see Set Up the Console Client [page 847].
- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Procedure

1. Open the command prompt and navigate to the folder containing `neo.bat/sh (<SDK installation folder>/tools)`.
2. Create the availability check.

   Execute:

   ```
   neo create-availability-check -a mysubaccount -b myhana:myhanaxsapp -u myuser
   -U /heartbeat.xsjs -C 6 -W 4 --host hana.ondemand.com
   ```

   - Replace **"mysubaccount"**, **"myhana:myhanaxsapp"** and **"myuser"** with the technical name of your subaccount, and the names of the productive SAP HANA database, application, and user respectively.
   - The availability URL (**/heartbeat.xsjs** in this case) is not provided by default by the platform. Replace it with a suitable URL that is already exposed by your SAP HANA XS application or create it. Keep in mind the limitations for availability URLs. For more information, see Availability Checks [page 767].

     > **i Note**
     >
     > In case you want to create an availability check for a protected SAP HANA XS application, you need to create a subpackage, in which to create an `.xsaccess` file with the following content:
     >
     > ```
     > {
     >     "exposed": true,
     >     "authentication": null,
     >     "authorization": null
     > }
     > ```
     >
     > In the SAP HANA XS Administration Tool (https://<productive SAP HANA name><subaccount>.<host>/sap/hana/xs/admin/), find the subpackage and activate public access to it. After this, you will be able to monitor the resources on and under the subpackage level, that is, you will be able to create an availability check for your protected application. Note that in the availability URL, you have to provide the path to the resource in the subpackage that you will be monitoring, for example **/heartbeat.xsjs**.

   - The check will trigger warnings **"-W 4"** if the response time is above **4** seconds and critical alerts **"-C 6"** if the response time is above 6 seconds or the application is not available.
   - Use the respective host for your subaccount type.

Related Information

Alerting [page 779]
Configure Availability Checks for SAP HANA XS Applications from the Cockpit [page 1006]

Regions and Hosts Available for the Neo Environment [page 17]

Availability Checks Commands

list-availability-check [page 1477]

create-availability-check [page 1383]

delete-availability-check [page 1401]

4.3.3.5 View Monitoring Metrics of a Database System

In the cockpit, you can view the current metrics of a selected database system to get information about its health state. You can also view the metrics history of a productive database to examine the performance trends of your database over different intervals of time or investigate the reasons that have led to problems with it. You can view the metrics for all types of databases.

Prerequisites

The *readMonitoringData* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Context

> **i Note**
>
> You can also retrieve the current metrics of a database system with the Metrics API.

Default Metrics of a Database System

All database systems include these default metrics. Other database-specific metrics may also be available.

Metric	Value	Execution Frequency
CPU Load	The percentage of the CPU that is used on average over the last minute.	This metric is updated every minute. An alert is triggered when 2 consecutive checks with an interval of 1 minute aren't in an OK state.
Disk I/O	The number of bytes per second that are currently being read or written to the disc.	This metric is updated every minute. An alert is triggered when 5 consecutive checks with an interval of 1 minute aren't in an OK state.

Metric	Value	Execution Frequency
Network Ping	The percentage of packets that are lost to the database host.	This metric is updated every minute. An alert is triggered when 2 consecutive checks with an interval of 1 minute aren't in an OK state.
OS Memory Usage	The percentage of the operating system memory that is currently being used.	This metric is updated every minute. An alert is triggered when 2 consecutive checks with an interval of 1 minute aren't in an OK state.
Used Disc Space	The percentage of the local discs of the operating system that is currently being used. **Note** If this metric is in a critical state, try restarting the database system. If the restart doesn't work, check the troubleshooting documentation. See the *Related Information* section.	This metric is updated every minute. An alert is triggered when 5 consecutive checks with an interval of 1 minute aren't in an OK state.

Default Metrics of an SAP HANA System

Metric	Value	Execution Frequency
HANA DB Availability	• *OK* - the database is reachable from our central admin component via JDBC. • *Critical* - either the database is down or overloaded, or there's a network issue.	This metric is updated every minute. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Alerting Availability	• *OK* - alerts can be retrieved from the SAP HANA system. • *Critical* - alerts can't be retrieved as there's no connection to the database. This also implies that any other visible metric may be outdated. This metric depends on the HANA DB Availability metric.	This metric is updated every minute. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Compile Server	• *OK* - the compiler server is running on the SAP HANA system. • *Critical* - the compile server crashed or was otherwise stopped. The service should recover automatically. If this doesn't work, a restart of the system might be necessary.	This metric is updated every 10 minutes. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.

Metric	Value	Execution Frequency
HANA DB Backup Volumes Availability	• *OK* - the backup volumes are available. • *Critical* - the backup volumes aren't available.	This metric is updated every 15 minutes. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Data Backup Age	• *OK* - the age of the last data backup is below the critical threshold. • *Critical* - the age of the last data backup is above the critical threshold.	This metric is updated every 24 hours. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Data Backup Exists	• *OK* - the data backup exists. • *Critical* - no data backup exists.	This metric is updated every 24 hours. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Data Backup Successful	• *OK* - the last data backup was successful. • *Critical* - the last data backup wasn't successful.	This metric is updated every 24 hours. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Log Backup Successful	• *OK* - the last log backup was successful. • *Critical* - the last log backup failed.	This metric is updated every 10 minutes. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA DB Service Memory Usage	• *OK* - no server is running out of memory. • *Critical* - a service is causing an out of memory error. See SAP Note 1900257.	This metric is updated every 5 minutes. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
HANA XS Availability	• *OK* - XSEngine accepts HTTPS connections. • *Critical* - XSEngine doesn't accept HTTPS connections.	This metric is updated every minute. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.

Metric	Value	Execution Frequency
HANA Dump Files Count	• *OK* - No dump files exist. • *Warning* - Up to 20 dump files exist. • *Critical* - More than 20 dump files exist. Try to analyze the dump files. **i Note** If you're still having issues, check the troubleshooting documentation. See the *Related Information* section.	The metric is updated every hour. An alert is triggered when a check isn't in an OK state.

Default Metrics of an SAP ASE System

Metric	Value	Execution Frequency
Sybase ASE Availability	• *OK* - the database is reachable from our central admin component via JDBC. • *Critical* - either the database is down or overloaded, or there's a network issue.	This metric is updated every minute. An alert is triggered when 3 consecutive checks with an interval of 1 minute aren't in an OK state.
Sybase ASE Long Running Trans	• *OK* - a transaction is running for up to an hour. • *Warning* - a transaction is running for more than an hour. • *Critical* - a transaction is running for more than 13 hours.	This metric is updated every 2 minutes. An alert is triggered when a consecutive check with an interval of 1 minute isn't in an OK state.
Sybase ASE HADR Fm State	FaultManager is a component for highly available (HA) SAP ASE systems that triggers a failover in case the primary node isn't working. • *OK* - FaultManager for a system that is set up as an HA system is running properly. • *Critical* - FaultManager isn't working properly and the failover doesn't work.	This metric is updated every 2 minutes. An alert is triggered when a consecutive check with an interval of 1 minute isn't in an OK state.
Sybase ASE HADR Latency	• *OK* - the latency for the HA replication path is less than or equal to 10 minutes. • *Warning* - the latency is greater than 10 minutes. • *Critical* - the latency is greater than 20 minutes. A high latency might lead to data loss if there's a failover.	This metric is updated every 2 minutes. An alert is triggered when a consecutive check with an interval of 1 minute isn't in an OK state.
Sybase ASE HADR Primary State	• *OK* - the primary host of a system that is set up as HA system is running fine. • *Critical* - the primary host isn't running properly.	This metric is updated every 2 minutes. An alert is triggered when a consecutive check with an interval of 1 minute isn't in an OK state.

Metric	Value	Execution Frequency
Sybase ASE HADR Standby State	• *OK* - the secondary or standby host of a system that is set up as HA system is running properly. • *Critical* - the secondary or standby host isn't running properly.	This metric is updated every 2 minutes. An alert is triggered when a consecutive check with an interval of 1 minute isn't in an OK state.

Procedure

1. Open the subaccount in the SAP BTP cockpit.
2. Navigate to the *Database Systems* page either by choosing *SAP HANA / SAP ASE* > *Database Systems* from the navigation area or from the *Overview* page.

   All database systems available in the selected subaccount are listed with their details, including the database version and state, and the number of associated databases.
3. Choose the entry for the relevant database system in the list.
4. Choose *Monitoring* from the navigation area to get detailed information about the current state and the history of metrics for the selected database system.

   When you open the checks history, you can view graphic representations for each of the different checks, and zoom in to see additional details. If you zoom in a graphic horizontally, all other graphics also zoom in to the same level of detail. Press `Shift` and drag to pan a graphic. Zoom out to the initial size by double-clicking.

   You can select different periods for each check. Depending on the interval you select, data is aggregated as follows:
   - Last 12 or 24 hours - data is collected every minute.
   - Last 7 days - data is aggregated from the average values for each 10 minutes.
   - Last 30 days - data is aggregated from the average values for each hour.

   You can also select a custom time interval for viewing check history.

Related Information

SAP BTP Cockpit

Metrics REST API for Database Systems [page 775]

4.3.3.6 Debug with SAP HANA Web-based Development Workbench

You can only debug SAP HANA server-side JavaScript with the SAP HANA Tools plugin for Eclipse as of release 7.4. If you are working with lower plugin versions, use the SAP HANA Web-based Development Workbench to perform your debugging tasks.

Prerequisites

- Your database user is assigned the *sap.hana.xs.debugger::Debugger* role.
- You have enabled debugging in the SAP HANA studio. Note that to do the following, you require the *sap.hana.xs.lm.roles::Administrator* role:
  1. Switch to the *Administration Console* perspective.
  2. Double-click your system.
  3. Switch to the *Configuration* tab.
  4. Set the following parameter: *xsengine.ini* > *debugger* > *enabled=true*. If the *debugger* section is not already present, create it and add the parameter *enabled*. Assign the value **`true`** to the *enabled* parameter.

Context

Procedure

1. In the SAP BTP cockpit, navigate to a subaccount. For more information, see Navigate in the Cockpit.
2. Choose *Applications* > *HANA XS Applications*.

   > **i Note**
   >
   > We recommend that you use the Google Chrome browser.

3. In the *HANA XS Applications* table, select the application to display its details.
4. In the *Application Details* section, click *Open in Web-based Development Workbench*. Note that the SAP HANA Web-based Development Workbench can also be opened directly at the following URL: `https://<database instance><subaccount>.<host>/sap/hana/xs/ide/`
5. Depending on whether you want to debug a `.xsjs` file or a more complex scenario (set a breakpoint in a `.xsjs` file and run another file), do the following:
   - `.xsjs` file:
     1. Set the breakpoints and then choose the *Run on server (F8)* button.

- Complex scenario:
  1. Set the breakpoint in the `.xsjs` file you want to debug.
  2. Open a new tab in the browser and then open the other file on this tab by entering its URL (`https://<database instance><subaccount>.<host>/<package>/<file>`).

  > **i Note**
  >
  > If you synchronously call the `.xsjs` file in which you have set a breakpoint and then open the other file in the SAP HANA Web-based Development Workbench and execute it by choosing the *Run on server (F8)* button, you will block your debugging session. You will then need to terminate the session by closing the SAP HANA Web-based Development Workbench tab.

> **i Note**
>
> If you leave your debugging session idle for some time once you have started debugging, your session will time out. An error in the WebSocket connection to the backend will be reported and your WebSocket connection for debugging will be closed. If this occurs, reopen the SAP HANA Web-based Development Workbench and start another debugging session.

4.3.3.7 Configure SAML 2.0 Authentication

Configure your HANA XS applications to use Security Assertion Markup Language (SAML) 2.0 authentication to implement identity federation with your corporate identity providers.

Context

The procedure that used to be described in this topic is deprecated. To configure SAML authentication, see Configure SSO with SAML Authentication for SAP HANA XS Applications.

4.3.3.8 Assign Trust Stores to HTTP Destinations

To be able to call SAP BTP services from SAP HANA XS applications, you need to assign a predefined trust store to the HTTP destination that defines the connection details for a specific service. The trust store contains the certificate required to authenticate the calling application.

Prerequisites

In the SAP HANA repository, you have created the HTTP destination (`.xshttpdest` file) to the service to be called. The file must have the `.xshttpdest` extension and be located in the same package as the application that uses it or in one of the application's subpackages.

Context

Procedure

1. Log on to the cockpit and choose *Applications* > *HANA XS Applications*.
2. In the *HANA XS Applications* screen, select the relevant application.
   The *Application Details* section lists all destinations created for the selected application, together with the subpackage name if the destination file is contained in a subpackage.
3. Select the relevant destination and choose *Assign Trust Store*.
4. In the dialog box, select the **DEFAULT** trust store from the dropdown box and save your entry. Note that **DEFAULT** specifies a predefined trust store containing the certificate required by SAP HANA XS applications.

Related Information

Maintaining HTTP Destinations

4.4 Multitarget Applications for the Neo Environment

A Multitarget Application (MTA) is a package comprised of multiple application and resource modules, which have been created using different technologies and deployed to different runtimes, but have a common lifecycle. You bundle the modules together, describe them along with their interdependencies to other modules, services, and interfaces, and package them in an MTA.

Complex business applications are composed of multiple parts developed with focus on micro-service design principles, API-management, usage of the OData protocol, increased usage of application modules developed with different languages, IDEs, and build methodologies. Thus, development, deployment, and configuration of separate elements introduce a variety of lifecycle and orchestration challenges. To address these challenges, SAP introduces the Multitarget Application (MTA) concept. It addresses the complexity of continuous deployment by employing a formal target-independent application model.

An MTA comprises of multiple modules created with different technologies, deployed to different target runtimes, but having a common lifecycle. Initially, developers describe the modules of the application, the interdependencies to other modules and services, and required and exposed interfaces. Afterward, the SAP BTP validates, orchestrates, and automates the deployment of the MTA.

For more information about the Multitarget Application model, see the official The Multitarget Application Model specification.

To learn more about	See
Multitarget Application deployment descriptor	Defining MTA Deployment Descriptors for the Neo Environment [page 1021]
Defining MTA Development Descriptors	Defining MTA Development Descriptors [page 1021]
Multitarget Application archive	Defining Multitarget Application Archives
Multitarget Application extension descriptor	Defining MTA Extension Descriptors
Multitarget Application module types and parameters	MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027]
How to model the Solution	Modeling Solutions [page 1062]
How to operate the Solution	Operating Solutions [page 1089]
How to use transport management tools for moving MTA archives among subaccounts	Integration with Transport Management Tools [page 1056]

Related Information

Multitarget Applications for the Neo Environment [page 1016]
JAR File Specification
Cloud MTA Build Tool

4.4.1 Create a Hello World Multitarget Application

In the context of the SAP BTP, a solution is comprised of:

- A Multitarget Application (MTA) archive that bundles all the deployable modules and configurations together with the accompanying MTA deployment descriptor, which describes the content of the MTA archive, the module interdependencies, and required and exposed interfaces
- An optional MTA extension descriptor that contains data complementary to the MTA deployment descriptor

In this tutorial, you will create the following:

1. A module that describes a Java application
2. A resource that describes your database binding
3. Additional metadata required for the deployment

Prerequisites

- You have created a Java application `.war` file.
- You have an SAP BTP database configured in your subaccount, and valid credentials.

Context

Procedure

1. Modeling the MTA deployment descriptor

   You define the entities that have to be deployed, namely modules and resources related to other modules, in the MTA deployment descriptor. This is a YAML file that defines the contract between you as a deployable artifact provider and the SAP BTP as a deployer tool. Initially, you have to describe the metadata required for the deployment.

   a. Create an empty text file and name it `mtad.yaml`.

   b. Using a text editor, enter the following data in the file:

   > **i Note**
   >
   > Strictly adhere to the correct indentations when working with YAML files, and do not use the tabulator character.

   > **Example**
   >
   > ```
   > _schema-version: '3.1'
   >
   > parameters:
   >   hcp-deployer-version: '1.1.0'
   >
   > ID: com.example.demo.basic
   > version: 0.1.0
   > ```

   The example above instructs the SAP BTP to:

   - Validate the Multitarget Application against the MTA specification version `3.1`
   - Use deploy features specific to the SAP BTP marked as version `1.0`
   - Deploy the Multitarget Application as a Solution with ID `com.example.demo.basic`
   - Consider the Multitarget Application version as a version `0.1.0`

   c. Create the module that describes the Java application. In the `mtad.yaml`, add the following data:

   > **Example**
   >
   > ```
   > modules:
   >   - name: example-java-app
   >     type: com.sap.java
   >     requires:
   >       - name: db-binding
   >     parameters:
   >        name: example
   >        jvm-arguments: -server
   >        java-version: JRE 7
   >        runtime: neo-java-web
   >        runtime-version: 1
   > ```

The example above instructs the SAP BTP to:

- Deploy a Java application with a specific runtime, Java version, and runtime arguments
- Require a MTA resource called `db-bindings`, where you describe your binding data

d. Describe the database binding `id` and the database credentials the Java application has to use by adding the following to the `mtad.yaml`:

> **Example**
>
> ```
> resources:
>   - name: db-binding
>     type: com.sap.hcp.persistence
>     parameters:
>       id:
> ```

The example above instructs the SAP BTP to create a database binding during the deployment process.

At this point of the procedure, no database `id` or credentials for your database binding have been added. The reason for this is that all the content of the `mtad.yaml` so far is a target-platform independent, meaning that the same `mtad.yaml` could be deployed to multiple SAP BTP subaccounts. The information about your database `id` and credentials are, however, subaccount-specific. To keep the `mtad.yaml` target platform independent, you have to create an MTA extension descriptor. This file is used in addition to your primary descriptor file, and contains data that is account-specific.

> **Note**
>
> Security-sensitive data, for example database credentials, should be always deployed using an MTA extension descriptor, so that this data is encrypted.

2. Create the MTA extension descriptor:
   a. Create an empty text file, and name it `dev.mtaext`.
   b. Using a text editor, enter the following data in the file:

> **Example**
>
> ```
> _schema-version: '3.1'
> ID: com.example.demo.basic.config
> extends: com.example.demo.basic
> parameters:
>   title: Basic Solution
>   description: This is a sample of a basic Solution.
>
> resources:
>   - name: db-binding
>     parameters:
>       id: dbalias
>       user-id: myuser
>       password : mypassword
> ```

The example above instructs the SAP BTP to:

- Extend the `com.example.demo.basic` MTA deployment descriptor with the `com.example.demo.basic.config` MTA extension descriptor
- Define the title and description for the target solution that will be visible in the SAP BTP
- Extend the `db-binding` resource and define within the following parameters:

- Alias of the database
- User for the database schema that you own
- Password for the database schema that you own.

At this point of the procedure, the MTA deployment descriptor, MTA extension descriptor, and the Java application `.war` files are compiled. You can now package your Multitarget Application archive and deploy it to the SAP BTP. The MTA archive contains all entities that have to be deployed. It also contains a manifest file, which links the modules and resources described into the MTA deployment descriptor file using their location in the archive. Using this data, the SAP BTP deploys the Multitarget Application archive as a solution.

3. Packaging your Multitarget Application
   a. Create an empty text file and name it `MANIFEST.MF`. Explicitly use upper-case letters.
   b. Using a text editor, enter the following data in the file:

> **Example**
>
> ```
> Manifest-Version: 1.0
> Created-By: example.com
> Name: example.war
> Content-Type: application/zip
> MTA-Module: example-java-app
> ```

   The example above instructs the SAP BTP to link the module `example-java-app` to the archive `example.war`.

> **Caution**
>
> Make sure that the `MANIFEST.MF` is compliant to the JAR file specification.

4. Create your Multitarget Application archive.

   MTA archives are compliant to the JAR file specification. This allows you to use commonly available tools for creating, modifying, and signing such archives. For the sake of the tutorial, we assume that you have a root directory named `/` where you place all the parts of the Multitarget Application before creating the archive.
   a. Create a folder called `META-INF` in the root directory.
   b. Place the `mtad.yaml` and the `MANIFEST.MF` files inside the `META-INF` directory.
   c. Place the `example.war` archive inside the root directory.

> **Note**
>
> The MTA extension descriptor file is deployed separately from the MTA archive.

   The root directory should now be structured as follows::

> **Example**
>
> ```
> /example.war
> /META-INF
> /META-INF/mtad.yaml
> /META-INF/MANIFEST.MF
> ```

   d. Archive the content of the root directory in an `.mtar` format using an archiving tool capable of producing a JAR archive.

Results

After you have created your Multitarget Application archive, you are ready to deploy it into the SAP BTP as a solution. To deploy the archive, proceed as described in Deploy a Standard Solution [page 1093].

4.4.2 Defining MTA Development Descriptors

Multitarget Applications are defined in a development descriptor required for design-time and build purposes.

The development descriptor (`mta.yaml`) defines the elements and dependencies of a Multitarget Application (MTA) compliant with the Neo environment.

> **i Note**
>
> The MTA development descriptor (`mta.yaml`) is used to generate the deployment descriptor (`mtad.yaml`), which is required for deploying an MTA to the target runtime. The MTA Archive Builder uses the MTA development descriptor in order to create an MTA archive, including the `mtad.yaml` and the `MANIFEST.MF` file.

An MTA development descriptor contains the following main elements, in addition to the deployment descriptor elements:

- `path`
- `build-parameters`

> **! Restriction**
>
> The WebIDE currently does not support creating MTA development descriptors for the Neo Environment. You have to create it manually by a text editor of your choice that supports the YAML serialization language.

For more information, see Multitarget Application Archive Builder.

4.4.3 Defining MTA Deployment Descriptors for the Neo Environment

The Multitarget Application (MTA) deployment descriptor is a YAML file that defines the relations between you as a provider of a deployable artifact and the SAP BTP as a deployer tool.

Using the YAML data serialization language, you describe the MTA in an MTA deployment descriptor (`mtad.yaml`) file containing the following:

- Modules and module types that represent Neo environment applications and content, which form the MTA and are deployed on the platform
- Resources and resource types that are not part of an MTA, but are required by the modules at runtime or at deployment time
- Dependencies between modules and resources

- Technical configuration parameters, such as URLs, and application configuration parameters, such as environment variables.

See the following examples of a basic MTA deployment descriptor that is defined in an `mtad.yaml` file:

Example

```
_schema-version: '3.1'
parameters:
  hcp-deployer-version: '1.1.0'

ID: com.example.descriptor
version: 0.1.0
modules:
  - name: example-java-app
    type: com.sap.java
    requires:
      - name: db-binding
    parameters:
       name: examplejavaapp
       jvm-arguments: -server
       java-version: JRE 7
       runtime: neo-java-web
       runtime-version: 1
resources:
  - name: db-binding
    type: com.sap.hcp.persistence
    parameters:
      id: fx7
      user-id:
      password:
```

The example above instructs the SAP BTP to:

- Deploy a Java application
- Create a database binding for that Java application

Note

- The format and available options in the MTA deployment descriptor could change with the newer versions of the MTA specification. Always specify the schema version when defining an MTA deployment descriptor, so that the SAP BTP is aware against which specific MTA specification version you are deploying.
- The example above is incomplete. In its case, you have to create an MTA extension descriptor containing the database user and password.
- As of `_schema version 3.1`, you have the option to input missing values that are required by the Multitarget Application, which afterwards act as the latest provided MTA extension descriptor. During deployment using the cockpit the SAP BTP detects the missing values, and opens a dialog where you can enter them. This option can be useful when you need to extend already provided MTAs with new data.
  For example, you can choose to provide credentials manually instead of storing and providing them in an MTA extension descriptor file. Also, you can manually input subaccount-relevant parameter values specific to the provider or consumer subaccount in the provider-consumer scenario. For more information, see the *Supported Metadata Options* subsection of MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].

Since the Neo environment supports a different set of module types, resource types, and configuration parameters, the deployment of an MTA archive can be further configured by using MTA extension descriptors.

This allows administrators to adapt a deployment to a target or use case specific requirements, like setting URLs, memory allocation parameters, and so on. For more information, see the official *Multitarget Application Model* specification.

Related Information

Defining MTA Extension Descriptors
Defining Multitarget Application Archives
MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027]
The Multitarget Application Model v.2
The Multitarget Application Model v.3

4.4.4 Defining Multitarget Application Archives

You package the MTA deployment descriptor and module binaries in an MTA archive. You can manually do so as described below, or alternatively use the Cloud MTA Build tool.

> **i Note**
>
> There could be more than one module of the same type in an MTA archive.

The Multitarget Application (MTA) archive is created in a way compatible with the JAR file specification. This allows us to use common tools for creating, modifying, and signing such types of archives.

> **⚠ Caution**
>
> The maximum size of an MTA archive is limited to 500 MB. Deployment is denied for archives with larger size.

An MTA archive consists of the following:

- The `MANIFEST.MF` file
- The `mtad.yaml` MTA deployment descriptor file
- Application binaries, content, or configuration files

> **i Note**
>
> - The MTA extension descriptor is not part of the MTA archive. During deployment you provide it as a separate file, or as parameters you enter manually when the SAP BTP requests them.
> - Using a `resources` directory as in some examples is not mandatory. You can store the necessary resource files on root level of the MTA archive, or in another directory with name of your choice.

The following example shows the basic structure of an MTA archive. It contains a Java application `.war` file and a `META-INF` directory, which contains an MTA deployment descriptor with a module and a `MANIFEST.MF` file.

Example

```
/example.war
/META-INF
/META-INF/mtad.yaml
/META-INF/MANIFEST.MF
```

The `MANIFEST.MF` file has to contain a name section for each MTA module part of the archive that has a file content. In the name section, the following information has to be added:

- `Name` - the path within the MTA archive, where the corresponding module is located. If it leads to a directory, add a forward slash (`/`) at the end.
- `Content-Type` - the type of the file that is used to deploy the corresponding module
- `MTA-module` - the name of the module as it has been defined in the deployment descriptor

Note

- You can store one application in two or more application binaries contained in the MTA archive.
- According to the JAR specification, there must be an empty line at the end of the file.

Sample content of the `META-INF/MANIFEST.MF` file:

Example

```
Manifest-Version: 1.0
Created-By: example.com
Name: example.war
Content-Type: application/zip
MTA-Module: example-java-app
```

The example above instructs the SAP BTP to:

- Look for the `example.war` file within the root of the MTA archive when working with module `example-java-app`
- Interpret the content of the `example.war` file as an `application/zip`

Note

The example above is incomplete. To deploy a solution, you have to create an MTA deployment descriptor. Then you have to create the MTA archive.

→ Tip

As an alternative to the procedure described above, you can also use the Cloud MTA Build Tool. See its official documentation at Cloud MTA Build Tool .

Related Information

https://sap.github.io/cloud-mta-build-tool/

The Multitarget Application Model v.2

The Multitarget Application Model v.3

JAR File Specification

Defining MTA Deployment Descriptors for the Neo Environment [page 1021]

Defining MTA Extension Descriptors

MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027]

4.4.5 Defining MTA Extension Descriptors

The Multitarget Application (MTA) extension descriptor is a YAML file that contains data complementary to the deployment descriptor. The data can be environment or deployment specific, for example, credentials depending on the user who performs the deployment. The MTA extension descriptor is a YAML file that has a similar structure to the deployment descriptor, by following the Multitarget Application Model structure with several limitations and differences. Normally, extension descriptor extends deployment descriptor but it is possible to extends other extension descriptor, making extension descriptors chain. It can add or overwrite existing data if necessary.

Several extension descriptors can be additionally used after the initial deployment.

> **Note**
>
> The format and available options within the extension descriptor may change with newer versions of the MTA specification. You must always specify the schema version option when defining an extension descriptor to inform the SAP BTP which MTA specification version should be used. Furthermore, the schema version used within the extension descriptor and the deployment descriptor should always be same.

In the examples below, we have a deployment descriptor, which has already been defined, and several extension descriptors.

> **Note**
>
> Each extension descriptor is defined in a separate file with an extension `.mtaext`.

Deployment descriptor:

> **Example**

```
_schema-version: '3.1'
parameters:
  hcp-deployer-version: '1.0'
ID: com.example.extension
version: 0.1.0
resources:
  - name: data-storage
    properties:
      existing-data: value
```

The example above instructs SAP BTP to:

- Validate the extension descriptor against the MTA specification version `3.1`
- Extend the `com.example.extension` deployment descriptor

The following is a basic example of an extension descriptor that adds data and overwrites data to another extension descriptor:

Example

```
_schema-version: '3.1'
ID: com.example.extension.first
extends: com.example.extension
resources:
  - name: data-storage
    properties:
      existing-data: new-value
      non-existing-data: value
```

The above instructs SAP BTP to:

- Extend the deployment descriptor by its ID `com.example.extension`
- Validate the extension descriptor against the MTA specification version 3.1
- Overwrite the data for the `existing-data` property
- Add a new data called `non-existing-data` to the `data-storage` properties

The following is an example of another extension descriptor that extends the extension descriptor from the previous example:

Example

```
_schema-version: '3.1'
ID: com.example.extension.second
extends: com.example.extension.first
resources:
  - name: data-storage
    properties:
      second-non-existing-data: value
```

The example above instructs the SAP BTP to:

- Extend the first extension descriptor by its ID
- Add a new data called `second-non-existing-data` to the `data-storage` properties
- The examples above are incomplete. To deploy a solution, you have to create a deployment descriptor and an MTA archive.

What is possible to do with an extension descriptor?

You can do the following using an extension descriptor:

- Add a new data in modules, resources, parameters, properties, provides, requires sections
- Overwrite an existing data (in depth) in modules, resources, parameters, properties, provides, requires sections
- As of schema version 3.xx, by default parameters and properties are overwritable and optional. If you want to make a certain parameter or property non-overwritable or required, you need to add specific metadata. SeeMetadata for Properties and Parameters.

You cannot use an extension descriptor to:

- Add new entities such as modules or resources
- Change module or resource type

- Alter read-only (system) parameters

Related Information

Defining MTA Deployment Descriptors for the Neo Environment [page 1021]
Defining Multitarget Application Archives
MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027]
The Multitarget Application Model v.2
The Multitarget Application Model v.3

4.4.6 MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment

> → Tip
>
> This section contains collapsible subsections. By clicking on the arrow-shaped icon next to a subsection, you can expand it to see additional information.

This section contains the parameters and options that can be used to compose the structure of an MTA deployment descriptor or an MTA extension descriptor.

> i Note
>
> As both descriptor types use the YAML file format, strictly adhere to the following syntax practices:
>
> - The parameter names are case-sensitive
> - The indents and spacing are specific

Supported Target Platform Options

The supported target platform options describe general behavior and information about the deployed Multitarget Application. The according options are placed within the primary part of the MTA deployment descriptor or MTA extension descriptor. That is, they are not placed within any modules or resources.

> i Note
>
> - Note that any sensitive data should be placed within the MTA extension descriptor.
> - To ensure that numeric values, such as product version and IDs, are not automatically interpreted as numbers, always wrap them in single quotes.

Option	Description	Type	Default Value	Mandatory
`_schema-version`	Version of the MTA specification to which the MTA deployment descriptor complies. The current supported versions by the SAP Business Technology Platform(SAP BTP) are: • `2.1` • `3.1`	Enclosed String, use single quotes	n/a	yes
`ID`	The identifier of the deployed artifact. The ID should follow the convention for a reverse-URL dot-notation, and it has to be unique within a particular subaccount.	String	n/a	yes
`extends`	Used in MTA extension descriptor to denote which MTA deployment descriptor should be extended. Applicable only in extension descriptors.	String	The ID of the deployment descriptor that is to be extended	yes
`version`	Version of the current Multitarget Application. The format of the version is a numeric string of **<major>.<minor>.<micro>** **i Note** The value must not exceed 64 symbols.	Enclosed String, use single quotes	n/a	yes
`parameters:` `hcp-deployer-version`	Version of the deploy service of the (SAP BTP) . This version differs from the schema version. The current supported versions are: • `1.0.0` • `1.1.0` • `1.2.0` **i Note** • Deployer version `1.0.0` is going to be deprecated. Use version `1.1.0` and higher. • During a solution update, a different technical approach is employed. For more information, see General Information About Solution Updates [page 1104].	Enclosed String, use single quotes	n/a	yes

Option		Description	Type	Default Value	Mandatory
	`title`	Human-readable title of the application.	The ID of the Multi-target Application	n/a	no
	`description`	Human-readable description of the application.	String	n/a	no
	`logo`	Base64-encoded logotype image. Optimize your image to be displayed for size 45x45 pixels. The supported formats are: • png • jpeg • gif The following syntax is for a `.png` logotype that has been encoded in Base64: **Example** `logo: " oAAABaCAMAAAAPdrEwAAAAnFBMVEX///..."`	String	n/a	no

Supported Multitarget Application Module Types

This section contains the modules that are supported by the SAP BTP and their parameters and properties.

Note

- The relation between a module and the entities created in the SAP BTP is not one-to-one, that is, it is possible for one module to contain several SAP BTP entities and vice versa.
- Any security-sensitive data, such as user credentials and passwords, has to be placed in the MTA extension descriptor.

Supported Module Types and Parameters

Tip

Expand the following subsections by clicking on the arrow-shaped element to see the available parameters and values.

The following module types are currently supported:

`com.sap.hcp.html5` - used for deploying HTML5 applications

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`name`	HTML5 application name, which has to be unique within the current subaccount.	String	n/a	yes
`display-name`	Human-readable name of the application. **i Note** The `display-name` and `name` parameters belong to an application level that is different from the one of the application versions. If another application version is defined in the MTA deployment descriptor, then its display name has to be identical to display names of other already defined versions of the application or has to be omitted.	String	n/a	no
`version`	Application version to be used in the HTML5 runtime. Used for deploying Java HTML5 modules with the same version can be deployed only once. In the `version` parameter, the usage of a `<timestamp>` read-only variable is supported. Thus, a new version string is generated with every deploy. For example, version: `'0.1.0-${timestamp}'`	String	n/a	yes
`active`	This flag indicates whether the related version of the application should be activated or not. The default value is `true`.	Boolean	`true`	no
`description`	Application description.	String	n/a	no
`subscribe`	When a provided solution is consumed, a subscription and designated entities might be created in the consumer subaccount, unless the parameter is set to `false`.	Boolean	`true`	no
`sfsf-access-point` (deprecated)	**⚠ Caution** The property is obsolete and therefore is no longer supported. If true, the application is activated for the SAP SuccessFactors system. The default value is `false`.	Boolean	`false`	no
`sfsf-idp-access`	If true, the extension application is registered as an authorized assertion consumer service for the SAP SuccessFactors system to enable the application to use the SAP SuccessFactors identity provider (IdP) for authentication.	Boolean	`false`	no

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`sfsf-home-page-tiles`	Registers SAP SuccessFactors Employee Central (EC) home page tiles in the SAP SuccessFactors company instance. This parameter is a YAML dictionary with one element with key `resource` and value `<path to resource>`. The resource is a descriptor file that defines the SAP SuccessFactors tiles. The resource has to be in JSON format. For more information, see Home Page Tiles JSON File [page 1198]. Ensure that each tile `name` is unique within the current subaccount.	Binary	n/a	no
`permissions`	HTML5 role assignment to specified permissions. The parameter is applicable for both a standard HTML5 application, as well as for subscribing to one as a consumer. • `name` - name of the permission • `role` - the role to which the permission is assigned to	List	n/a	no

`com.sap.java` and `java.tomcat` - used for deploying Java applications, either with the proprietary SAP Java Web or the Java Web Tomcat runtime containers.

For more information about runtime containers, see Application Runtime Container [page 857].

> **i Note**
>
> You can deploy these application types using two or more `war` files contained in the MTA archive.

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`name`	Java application name, which has to be unique within the current subaccount. The `name` value length has to be between 1 and 255 symbols.	String	n/a	yes

`runtime`	Depending on the module and its used runtime, use one of the following:
	
	• For `com.sap.java`
	  ○ `neo-java-web`
	  ○ `neo-javaee6-wp`
	  ○ `neo-javaee7-wp`
	• For `java.tomcat` - do not define this parameter
		String	`neo-java-web`	yes

Supported Parameter	Parameter Description	Type	Default Value	Mandatory

`runtime-version`	If defining a specific runtime version is required, use one of the following:
	• For `com.sap.java` - for example, `1` or `2`
	• For `java.tomcat` - for example, `2` or `3`. The major supported runtime versions are `2` (with Tomcat 7) and `3` (with Tomcat 8).
		Enclosed String, use single quotes	• For `com.sap.java`
			  ○ For `neo-java-web-1`
			  ○ For `neo-javaee6-wp-2`
			• For `java.tomcat-2`
				no

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`java-version`	The JVM major version, for example `JRE 7` or `JRE 8`.	String	`JRE 7`	no
`compute-unit-size`	The virtual machine computing unit size. The available sizes are `LITE`, `PRO`, `PREMIUM`, `PREMIUM_PLUS`. For more information, see Compute Units [page 867].	String	`LITE`	no
`minimum-processes`	Minimum number of process instances. The allowed range is from `1` to `99`. **i Note** You either have to use both the `minimum-processes` and `maximum-processes` parameters, or neither.	Integer	`1`	no
`maximum-processes`	Maximum number of process instances. The allowed range is from `1` to `99`. **i Note** • You either have to use both the `minimum-processes` and `maximum-processes` parameters, or neither. • The `maximum-processes` should be equal to or higher than the `minimum-processes` value.	Integer	`1`	no
`rolling-update`	Performs update of an application without downtime in one go. **i Note** At least `hcp-deployer-version` 1.2.0 is required.	Boolean	`false`	no
`rolling-update-timeout`	Defines how long the old process will be disabled before it is stopped. **i Note** At least `hcp-deployer-version` 1.2.0 is required.	Integer	`60`	no
`running-processes`	Specifies how many processes will run at the end of the state of the Java application. If not specified, the minimum number is used.	Integer	n/a	no
`jvm-arguments`	The relevant JVM arguments employed by the customer application.	String	n/a	no
`max-threads`	The maximum allowed number of threads.	Integer	`200`	no
`connection-timeout`	The maximum timeout period for the connection, in milliseconds.	Integer	`20000`	no
`encoding`	The used Uniform Resource Identifier (URI) encoding standard.	String	`ISO-8859-1`	no

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`compression`	The use of gzip compression for optimizing HTTP response time between the Web server and its clients. The available values are `"on"`, `"off"`, `forced`. **i Note** • Always wrap `"on"` or `"off"` values in quotation marks. • Explicitly specify the `compression-mime-types` and `compression-min-size` parameters only when you use the value `"on"`.	String	`"off"`	no
`compression-mime-types`	The used compression mime type, for example `text/json text/xml text/html`	String	n/a	no
`compression-min-size`	The threshold size above which an HTTP response package is compressed to reduce traffic.	Integer	n/a	no
`role-provider`	Defines the application that provides the role for the Java application. Use one of the following: • `sfsf` • `hcp`	String	n/a	no
`roles`	Maps predefined Java application roles to the groups they have to be assigned to. It has to specify the following parameters: • `name` - its value has to be the name of the role • `groups` - its value has to be a list of group names to which the role is assigned to	List	n/a	no
`subscribe`	When a provided solution is consumed, a subscription and designated entities might be created in the consumer subaccount, unless the parameter is set to `false`.	Boolean	`true`	no
`sfsf-access-point` (deprecated)	**⚠ Caution** The property is obsolete and therefore is no longer supported. If true, the application is activated for the SAP SuccessFactors system. The default value is `false`.	Boolean	`false`	no
`sfsf-idp-access`	If true, the extension application is registered as an authorized assertion consumer service for the SAP SuccessFactors system to enable the application to use the SAP SuccessFactors identity provider (IdP) for authentication.	Boolean	`false`	no

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`sfsf-connections`	Use this to configure the connectivity of a Java extension application to an SAP SuccessFactors system. It creates the required HTTP destination and registers an OAuth client for the Java application in SAP SuccessFactors. **i Note** Note that SFSF connections can only be created after the corresponding Java application has been deployed and started. This means that an `sfsf-connections` module depends on a `com.sap.java` module. - You can create connections to one or more applications, and employ one or more connection types. - Registers the SAP SuccessFactors Employee Central (EC) home page. Note that existing SAP Fiori custom roles are skipped during deployment. The `sfsf-connections` parameter is a YAML list comprised of entries with the following attributes: - `type` - values can be `default` or `technical-user`. Note that if you choose `default`, you cannot define a `technical-user-id`. - `sfsf-module` - possible vlaues are `BizX` (default value) or `LMS` (deprecated). Use this parameter to configure a connection to the SAP SuccessFactors BizX Suite or SAP SuccessFactors Learning Management System (LMS) (deprecated), respectively. **⚠ Caution** The `LMS` value of the `sfsf-module` attribute is no longer supported. You can configure connections to LMS as described in Background Knowledge for SAP SuccessFactors Learning Web Services. - `additional-properties` - any additional properties defining the connection, specified as name-value pairs. - `technical-user-id` - if you use the `technical-user` value for the `type` parameter, you have to define a technical user ID. You can create more than one SAP SuccessFactors connections. For a comprehensive example of a `sfsf-connections`, see Modeling SAP SuccessFactors Extensions [page 1083].	List	n/a	no

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`sfsf-outbound-connections` (deprecated)	**⚠ Caution** This property is no longer supported. You can still configure the connectivity from SAP SuccessFactors to applications deployed in SAP BTP, Neo environment. See Integrate the Extension Application with SAP SuccessFactors Intelligent Services. Configures the connectivity from a SAP SuccessFactors system to the Java application. It creates the required OAuth client to the Java application and, if required, the application identity provider configuration. The `sfsf-outbound-connections` parameter is a YAML list comprised of entries with the following attributes: • `type` - configures the authentication type of the connection. It can be `OAuth2SAMLBearerAssertion` or `OAuth2ClientCredentials` • `name` - your preferred name for the outbound connection you are creating • `subject-name-id` - the name of the attribute you want to use as subject ID • `subject-name-id-format` - the format of the attribute you want to use as subject ID • `assertion-attributes-mapping` - the mapping of assertion attributes you want to define in the application identity provider for this outbound connection, specified as name-value pairs. For a comprehensive example of a `sfsf-outbound-connections`, see Modeling SAP SuccessFactors Extensions [page 1083].	List	n/a	no
`sfsf-home-page-tiles`	Registers SAP SuccessFactors Employee Central (EC) home page tiles in the SAP SuccessFactors company instance. This parameter is a YAML dictionary with one element with key `resource` and value `<path to resource>`. The resource is a descriptor file that defines the SAP SuccessFactors tiles. The resource has to be in JSON format. For more information, see Home Page Tiles JSON File [page 1198]. Ensure that each tile `name` is unique within the current subaccount.	Binary	n/a	no

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`destinations`	This parameter is a YAML list comprised of one or more connectivity destinations. To see the available parameters and values, see the table "Destination Parameters" below. **Note** • If you have sensitive data, all destination parameters have to be moved to the MTA extension descriptor. • When you redeploy a destination, any parameter changes performed after deployment of the destination are removed. Your custom changes have to be performed again.	List	n/a	no

`com.sap.fiori` - used for deploying SAP Fiori configurations

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`owner`	Indicates in which subaccount the content should be imported. The possible values are `provider` or `consumer`.	String	`provider`	no
`mta-id`	The same as the global ID. **Note** • To reduce the risk of being out of sync, we recommend that you use YAML anchors. • The value must not exceed 64 symbols.	String	n/a	no
`mta-version`	The same as the global version. **Note** • To reduce the risk of being out of sync, we recommend that you use YAML anchors. • The value must not exceed 64 symbols.	Enclosed String **Note** Use single quotes.	n/a	no

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`back-end-ppms-product-version-id`	ID of the PPMS product version backend.	Enclosed String **i Note** Use single quotes.	n/a	no
`back-end-ppms-support-package-stack-id`	ID of the PPMS support-package stack backend	Enclosed String **i Note** Use single quotes.	n/a	no
`back-end-cross-product-stack-id`	ID of the cross product stack backend	Enclosed String **i Note** Use single quotes.	n/a	no

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`back-end-cross-product-stack-solution-id`	ID of the cross product stack solution backend	Enclosed String **i Note** Use single quotes.	n/a	no
`release`	If it is set to `true`, it marks a released product.	Boolean	`false`	no
`target-site-id`	It specifies the target site in which the content will be deployed.	String	n/a	no
`minimum-sapui5-version`	Version of the minimum required SAPUI5 Runtime. The format of the version is a numeric string of `<major>.<minor>` or `<major>.<minor>.<micro>`	Enclosed String **i Note** Use single quotes.	n/a	no

> **i Note**
>
> You have to ensure that the `back-end-*-id` parameter values are numeric strings of exactly 20 digits.

`com.sap.fiori.app` - used for deploying SAP Fiori applications

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`html5-app-name`	SAP Fiori application name, which has to be unique within the current subaccount.	String	n/a	yes

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`html5-app-display-name`	Human-readable name of the application. **i Note** The `html5-app-display-name` and `html5-app-name` parameters belong to an application level that is different from the one of the application versions. If another application version is defined in the MTA deployment descriptor, then its display name has to be identical to display names of other already defined versions of the application or has to be omitted.	String	n/a	no
`html5-app-version`	SAP Fiori application version **i Note** The same rules apply as for the `sap.com.hcp.html5 version` parameter with the difference that this parameter is optional. Default value: `'${timestamp}'`	String	`${timestamp}`	no
`html5-app-active`	This flag indicates whether the related version of the application should be activated or not. The default value is `true`.	Boolean	`true`	no
`html5-app-permissions`	HTML5 role assignment to specified permissions. The parameter is applicable for both an HTML5 standard application, as well as for subscribing as a consumer. • `name` - name of the permission • `role` - the role to which the permission is assigned to	List	n/a	no

`com.sap.fiori.role` - used for deploying custom SAP Fiori roles

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`name`	SAP Fiori custom role name, which has to be unique within the current subaccount. The `name` value length has to be between 1 and 255 symbols.	String	n/a	yes
`groups`	List of group names to which the role has to be assigned.	List	n/a	no

For more information, see Role Assignment of Fiori Roles to Security Groups [page 1088].

`com.sap.hcp.html5.role` - used for deploying custom HTML5 application roles

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`name`	HTML5 application custom role name, which has to be unique within the current subaccount. The `name` value length has to be between 1 and 255 symbols.	String	n/a	yes
`groups`	List of group names to which the role has to be assigned.	List	n/a	no

For more information, see Role Assignment of HTML 5 Roles to Security Groups [page 1089].

`com.sap.odp.config` - used for deploying OData Provisioning configuration

> → Remember
>
> The use of this module type with parameters valid for `hcp-deployer-version: '1.0.0'` will soon be de-supported. We recommend that you use the parameters valid for `hcp-deployer-version: '1.1.0'`, or adapt your module type accordingly.

- Supported parameters when used with `hcp-deployer-version: '1.0.0'`

  > → Remember
  >
  > This deployer version will soon be de-supported. We recommend you use `1.1.0`.

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`metadata-validation-setting`	Enable or disable metadata validation, for example `true`.	Boolean	n/a	yes
`metadata-cache-setting`	Enable or disable metadata cache, for example `false`.	Boolean	n/a	yes

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`services`	List of OData services. Parameters required for an OData service are: ○ `name` - name of the service from the SAP on-premise back-end system that needs to be registered, for example `EPM_DEVELOPER_SCENARIO_SRV`. This name has to be unique within the current subaccount. ○ `namespace` - namespace for the above service being registered from the SAP on-premise back-end system, for example `IWBEP` ○ `version` - version of service being registered from the SAP on-premise back-end system, for example `1` ○ `description` - description of the service being registered, for example `/IWBEP/CL_EPM_DEVELOPE_DPC_EXT` ○ `initial-status` - status of the service registered in the OData Provisioning Administration cockpit that is set during the initial deployment of the OData service, whether it is active or inactive, for example `true` ○ `overwrite-existing` - this flag declares whether an existing OData service is going to be updated or not, for example `false` ○ `model-id` - the model ID corresponding to the metadata of the service being registered, for example `/IWBEP/EPM_DEVELOPER_SCENARIO_MD_0001` ○ `initial-default-destination` - the SAP on-premise back-end system destination where the service exists that is set during the initial deployment of the OData service. `Default` means that if the service does not specify any origin in the URL, then the metadata and data requests are going to be fulfilled from the destination mentioned against `initial-default-destination`, for example `AP2`.	List	n/a	yes

- Supported parameters when used with `hcp-deployer-version: '1.1.0'`:

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`metadata-validation-setting`	Enable or disable metadata validation, for example `true`.	Boolean	n/a	yes
`metadata-cache-setting`	Enable or disable metadata cache, for example `false`.	Boolean	n/a	yes

Supported Parameter	Parameter Description	Type	Default Value	Mandatory
`services`	List of OData services. Parameters required for an OData service are: ○ `name` - name of the service from the SAP on-premise back-end system that needs to be registered, for example `EPM_DEVELOPER_SCENARIO_SRV`. This name has to be unique within the current subaccount. ○ `namespace` - namespace for the above service being registered from the SAP on-premise back-end system, for example `IWBEP` ○ `version` - version of service being registered from the SAP on-premise back-end system, for example `1` ○ `status` - status of the service registered in the OData Provisioning Administration cockpit. To make the service active use `true`, or `false` to keep it inactive. ○ `initial-description` – description of the service. For example `/IWBEP/CL_EPM_DEVELOPE_DPC_EXT`. Use only when the service is created for the first time. **i Note** If a service with the same `name/namespace/version` combination already exists but has different description, the import will fail. ○ `initial-model-id` - the model ID corresponding to the metadata of the service, for example `/IWBEP/EPM_DEVELOPER_SCENARIO_MD_0001`. Use only when the service is created for the first time. **i Note** If a service with the same `name/namespace/version` combination already exists but has different `model-id`, the import will fail. ○ `initial-default-destination` - the name of the default SAP on-premise back-end system destination that is going to be used by the OData Provisioning service when no-multi origin composition (MOC) is specified as a parameter in the OData URL. Use only when the service is created for the first time. **i Note** If a service with the same `name/namespace/version` combination already exists but has different default destination, the import will fail.	List	n/a	yes

`com.sap.hcp.sfsf-roles` - used for uploading and importing SAP SuccessFactors HCM Suite roles from the SAP BTP system repository into the SAP SuccessFactors customer instance.

The role definitions must be described in a JSON file. For more information about creating `roles.json` file, see Create the Resource File with Role Definitions [page 1202].

Ensure that each role has a unique `roleName` within the current subaccount.

`com.sap.hcp.group` **- used for modeling the SAP BTP authorization groups.**

Supported Parameter	Parameter Description	Type	Default	Mandatory
`name`	Group name, which has to be unique within the current subaccount. The `name` value length has to be between 1 and 255 symbols.	String	n/a	yes

`com.sap.hcp.destination` **- A generic subaccount-level destination.**

To see the available parameters and values, see the table "Destination Parameters" below.

`com.sap.hcp.portal.destination` **- A service-level destination for theSAP Cloud Portal service.**

To see the available parameters and values, see the table "Destination Parameters" below.

`com.sap.hcp.odp.destination` **- A service-level destination for the SAP OData Provisioning.**

To see the available parameters and values, see the table "Destination Parameters" below.

`com.sap.integration` **- used for modeling the content for the SAP Cloud Integration.**

Supported Parameter	Parameter Description	Type	Default	Mandatory
`resource-type`	Type of the resource	String	n/a	yes
`resource-id`	ID of the resource	String	n/a	yes
`version`	Version of the `com.sap.integration` module type	String	n/a	yes
`technical-name`	Technical name of the `com.sap.integration` module type	String	n/a	yes

> **i** Note
>
> To use the `com.sap.integration` module type, first you have to:
>
> - Enable the SAP Solution Lifecycle Management service for SAP BTP service in a subaccount that supports SAP Cloud Integration. For more information, see Content Transport in the SAP Cloud Integration documentation.
> - Create a destination with named `CloudIntegration` with the following properties:
>   - Type - **`HTTP`**
>   - URL - URL pointing to the `/itspaces` of the TMN node for the SAP Cloud Integration tenant in the current subaccount
>   - Proxy Type - **`Internet`**
>   - Authentication - **`BasicAuthentication`**
>   - User and password - credentials of a user that has the `AuthGroup.IntegrationDeveloper` role for the above-mentioned TMN node
>
> For more information, see Using Services in the Neo Environment [page 1152].

Supported Multitarget Application Resource Types

This section contains the resource types and their parameters that are supported by the SAP BTP.

> **Note**
>
> - The relation between a module and the entities created in the SAP BTP is not one-to-one, that is, it is possible for one module to contain several SAP BTP entities.
> - Any security-sensitive data, such as user credentials and passwords, has to be placed in the MTA extension descriptor.

Resource Type	Description
`com.sap.hcp.persistence`	Used for binding a database with a specified identifier to a Java application
`<untyped>`	Used for adding any properties that you might require and which you define. It does not have a lifecycle. **Note** The untyped resource is unclassified, that is, it does not have a type.

Resource types support the following parameters:

Resource type	Parameter	Parameter Description	Type	Default	Mandatory
`com.sap.hcp.persistence`	`id`	Identifier of the database that will be bound to a deployed Java application You can model a named data source by using the parameter **Note** If you want to use a `<DEFAULT>` database binding, the standard data source `jdbc/DefaultDB` has to be set up at the stage of the Java application development.	String	n/a	yes
	`user-id`	The user ID parameter allows a database to be bound to a preexisting schema. If not used, a new schema has to be created. **Note** We recommend that you place this parameter in the MTA extension descriptor, if you are using one.	String	n/a	no

Resource type	Parameter	Parameter Description	Type	Default	Mandatory
	`password`	You can model a named data source by using the parameter. The password parameter allows a database to be bound to a preexisting schema. If not used, a new schema has to be created. **i Note** We recommend that you place this parameter in the MTA extension descriptor, if you are using one.	String	n/a	no
	`account`	This is the provider subaccount of the database. The subaccount parameter is needed only when the database is consumed from another subaccount. **i Note** The provider subaccount must meet the following criteria: • It has to be part of the same global account. • It has to provide permissions that allow a database to be used from another subaccount. For more information, see Add New Access Permissions.	String	n/a	no

`binding-name` that is added to the database binding resource required in the `requires` section of the `com.sap.java` and `java.tomcat` module types.

Required Sections	Parameter	Parameter Description	Type	Default	Mandatory
`com.sap.java` and `java.tomcat`	`binding-name`	Data source name **i Note** If you want to use a named database binding, the corresponding data source `jdbc/<name>` has to be set up at the stage of the Java application development. For more information, see Modeling Database Bindings [page 1066].	String	n/a	yes

Supported Metadata Options

The MTA specification `_schema-version 3.1` introduces the notion for metadata, which can be added to a certain property or parameter.

Metadata Option	Description	Type	Default Value	Mandatory

`consumer-optional`	Used when you want to provide your Multitarget Application for consumption by other subaccounts. You can add the `consumer-optional` metadata to a property to indicate that it should be populated with an MTA extension descriptor when your subscribers consume the Multitarget Application. If you do not provide the `consumer-optional` metadata, the deployment of the MTA deployment descriptor within your subaccount will fail due to missing data.
	
	The following example will require the consumer of your Multitarget Application to provide a user and a password value:
	
	Example
	`resources:`
	`  - name: example-resource`
	`    properties:`
	`      user:`
	`      password:`
	`    properties-metadata:`
	`      user:`
	`        optional: true`
	`        consumer-optional: false`
	`      password:`
	`        optional: true`
	`        consumer-optional: false`
	`    ...`
	
	Note
	• The `optional` parameter has to be explicitly defined and set to `true` if you want to use the option `consumer-optional`. See the MTA specification for additional information.
	• This option is available for Multitarget Application schema 3.1.0 and higher
		Boolean	`true`	no

Metadata Option	Description	Type	Default Value	Mandatory

`description`	If it is used as metadata about a parameter without a value, it provides a user-friendly description of the missing parameter in the cockpit.
	
	Example
	`resources:`
	`  - name: example-resource`
	`  properties:`
	`  user:`
	`  properties-metadata:`
	`  user:`
	`  description: Example resource`
	`user name`
	`  ...`
		String		no

`sensitive`	If it is used as metadata about a parameter without a value, it prompts the cockpit to use a password input field, that is, with hidden content.
	
	Example
	`resources:`
	`  - name: example-resource`
	`  properties:`
	`  password:`
	`  properties-metadata:`
	`  password:`
	`  sensitive: true`
	`  ...`
		Boolean	`false`	no

`complex`	If it is used as metadata about a parameter without a value, it prompts the cockpit to use an input field for free text, for example, a description of a solution in several lines of text.
	
	Example
	`resources:`
	`  - name: example-resource`
	`  properties:`
	`  description:`
	`  properties-metadata:`
	`  description:`
	`  complex: true`
	`  ...`
	
	Note
	This parameter is not taken into account if you use it in conjunction with the `sensitive` parameter. The *Password* input field is used instead.
		Boolean	`false`	no

Metadata Option	Description	Type	Default Value	Mandatory

`default-value`	If it is used as metadata about a parameter without a value, it provides a cockpit-relevant default value of the missing parameter.
	Example
	`resources:`
	`  - name: example-resource`
	`  properties:`
	`  user:`
	`  properties-metadata:`
	`  user:`
	`  default-value: John Doe`
	`  ...`
		String		no

Destination Parameters

Depending on the type of the destination that you wish to create (subaccount-level, application-level, subscription destination, and so on), the destination can be modeled as a module `com.sap.hcp.destination`, or as a parameter of the modules `com.sap.java` or `java.tomcat`. However, the options available when you create a destination are the same for all of the destination types.

MTA Parameter	Type	Mandatory	Possible Values	Default Value	Description or Comment
`force-overwrite`	String		`true` or `false`	`false`	If `true`, an already existing destination with the same name will be overwritten.
`owner`	String			`provider`	Indicates in which subaccount the destination should be created. The possible values are `provider` or `consumer`.
`name`	String	Yes		n/a	Technical name of the destination. It can be used later on to get an instance of that destination programmatically.
`type`	String	Yes	`HTTP, LDAP, MAIL, RFC`	HTTP	
`description`	String				
`url`	URL	Yes			Use when the parameter `type` has the `HTTP` or `LDAP` values. Mandatory only for these types.
`proxy-type`	Enumeration		`Internet, OnPremise`	Internet	Use with `HTTP` or `LDAP` destination `type`.
`authentication`	Enumeration		`NoAuthentication, BasicAuthentica`	`NoAuthentication`	Use with `HTTP` or `LDAP` destination `type`.

MTA Parameter	Type	Mandatory	Possible Values	Default Value	Description or Comment
			`tion`, `AppToAppSSO`, `ClientCertificateAuthentication`, `OAuth2SAMLBearerAssertion`, `PrincipalPropagation`, `SAPAssertionSSO`		
`user`	String				When to use: • if `BasicAuthentication` is the `Authentication` type. Mandatory only for this type. • if the parameter `type` has the values `MAIL`, or `RFC`.
`password`	String				When to use: • if `BasicAuthentication` is the `Authentication` type. Mandatory only for this type. • if `MAIL`, or `RFC` are the destination `type`.
`audience`	String	Yes			Use when `OAuth2SAMLBearerAssertion` is the `Authentication` type. Mandatory only for this authentication.
`client-key`	String	Yes			Use when `OAuth2SAMLBearerAssertion` is the `Authentication` type. Mandatory only for this authentication.
`token-service-url`	URL	Yes			Use when `OAuth2SAMLBearerAssertion` is the `Authentication` type. Mandatory only for this authentication.
`token-service-user`	String				Use when `OAuth2SAMLBearerAssertion` is the `Authentication` type.
`token-service-password`	String				Use when `OAuth2SAMLBearerAssertion` is the `Authentication` type.
`system-user`	String				Use when `OAuth2SAMLBearerAssertion` is the `Authentication` type.

MTA Parameter	Type	Mandatory	Possible Values	Default Value	Description or Comment
`issuer-sid`	String	Yes			Use when `SAPAssertionSSO` is the `Authentication` type. Mandatory only for this type.
`issuer-client`	String	Yes			Use when `SAPAssertionSSO` is the `Authentication` type. Mandatory only for this type.
`recipient-sid`	String	Yes			Use when `SAPAssertionSSO` is the `Authentication` type. Mandatory only for this type.
`recipient-client`	String	Yes			Use when `SAPAssertionSSO` is the `Authentication` type. Mandatory only for this type.
`certificate`	String	Yes			Use when `SAPAssertionSSO` is the `Authentication` type. Mandatory only for this type.
`signing-key`	String	Yes			Use when `SAPAssertionSSO` is the `Authentication` type. Mandatory only for this type.
`system-user`	String				Use when `OAuth2SAMLBearerAssertion` or `SAPAssertionSSO` are the `Authentication` type.
`repository-user`	String				Use with the `RFC` parameter `type`.
`repository-password`	String				Use with the `RFC` parameter `type`.
`client`	String	Yes	3 digits, in single quotes		Use with the `RFC` parameter `type`. Mandatory only for this type.
`client-ashost`	String		00-99, in single quotes		Use with the `RFC` parameter `type`. Either this or `client-mshost` must be specified.
`client-sysnr`	String				Use with the `RFC` parameter `type`, if `client-ashost` is specified.
`client-mshost`	String				Use with the `RFC` parameter `type`. Either this or `client-ashost` must be specified.
`client-r3name`	String		3 letters or digits		Use with the `RFC` parameter `type`, if `client-mshost` is specified.
`client-msserv`	String				Use with the `RFC` parameter `type`., if `client-mshost` is specified.
`additional-properties`	Map				Use this section to define arbitrary properties as a map of keys and values

MTA Parameter	Type	Mandatory	Possible Values	Default Value	Description or Comment

					used in the destination. and are also validated by the SAP BTP. The additional parameters are denoted as follows:
					Example
					`ldap.`
					`mail.`
					`jco.client.`
					`jco.destination.`

`location-id`	String				The value defines the location ID identifying the Cloud Connector over which the connection is opened. The default value is an empty string identifying the Cloud Connector that is connected without any location ID.

Note

The `additional-properties` values are not strictly verified during deployment, since they may vary excessively. For example, such values might depend on the destination type or authentication type. In case you are using such additional values, after deployment you have to ensure that the required elements have been properly created, and they operate as expected.

Destinations Target Platform Keyword Options

In regard to modeling destinations the SAP BTP offers several keyword properties, which you can use to optimize your declaration about deploying a destination. You can have the following destination types:

Keyword Option	Description

`application-url`	This keyword can be placed only within the properties category of the `provides` section of the `com.sap.java` and `java.tomcat` module types. It is used when you want to extract the URL of the Java Application and link it to a destination that you have modeled.
	
	The following example contains a Java application that has a destination that leads to itself. Note that this example uses the MTA placeholder concept. For more information, see "Destination with Specific Target Platform Data Options" below.
	
	Example
	
	`modules:`
	`  - name: java-module`
	`  type: com.sap.java`
	`  provides:`
	`  - name: java-module`
	`  properties:`
	`  application-url: ${default-url}`
	`  requires:`
	`  - name: java-module`
	`  parameters:`
	`  name: exampleapp`
	`  destinations:`
	`  - name: ExampleWebsite`
	`  type: HTTP`
	`  url: ${java-module/application-url}`
	`  ...`

Destinations with Target Platform Specific Data Options

When modeling destinations the SAP BTP offers several keyword properties that allow you to express your intention when deploying a destination more clearly. There might be cases, in which some of the destination data prior deploying the MTA archive is not known to you. Such data might be, for example, the URL of a Java Application that you want your destination to point to. To address these cases, SAP BTP provides several placeholders that you can use when you model your MTA. Placeholders are part of the Multitarget Application specification, and are strings resolved depending on the scope in which they are used. They have the syntax `${<name>}`.

Currently all types of destinations support the following placeholders, which are automatically resolved with their valid values during deployment.

Placeholder Option	Description

`${default-url}`	Instructs SAP BTP to resolve the placeholder value to the default Java Application URL when deploying the destination. This placeholder can be part only of the property named `application-url`, which serves as a `provided` dependency of the `com.sap.java` and `java.tomcat` module types.
	
	This example shows the usage of the `${default-url}` placeholder. The modeled java-module provides the `application-url` dependency:
	
	Example
	`modules:`
	`  - name: java-module`
	`  type: com.sap.java`
	`  provides:`
	`  - name: java-module`
	`  properties:`
	`  application-url: ${default-url}`
	`  parameters:`
	`  name: exampleapp`
	`  ...`
	
	Note
	• This placeholder can be used only with destination types that have a URL within their properties, that is, destination types such as an HTTP destination.
	• This URL can be automatically resolved only if the Java Application has only one URL.

`${account-name}`	Instructs SAP BTP to resolve the placeholder value to your subaccount name when deploying the destination. This placeholder can be used only in the `url` parameter for a destination, the `token-service-url` parameter, and in the `application-url` property, which serves as a `provided` dependency of the `com.sap.java` and `java.tomcat` module types.
	
	Example
	`modules:`
	`  - name: java-module`
	`  type: com.sap.java`
	`  provides:`
	`  - name: java-module`
	`  properties:`
	`  application-url: ${default-url}/accounts/${account-name}/example`
	`  parameters:`
	`  name: exampleapp`
	`  ...`
	`  - name: abc-java`
	`  type: com.sap.java`
	`  parameters:`
	`  destinations:`
	`  - name: ExampleWebsite`
	`  type: HTTP`
	`  url: http://abc.example.com/accounts/${account-name}`
	`  ....`

Placeholder Option	Description

`${provider-account-name}`	Instruct SAP BTP to resolve the placeholder value to the subaccount name of the provider when the destination is being deploying. This placeholder can be used only in the `url` parameter for a destination and the `token-service-url` parameter. You can use it if you want to employ a model, where a destination is created within your subscribers subaccount and you want it to point to a URL in your provider subaccount.
	
	Example
	`modules:`
	`  - name: abc-java`
	`  type: com.sap.java`
	`  parameters:`
	`  destinations:`
	`  - name: ExampleWebsite`
	`  type: HTTP`
	`  url: http://abc.example.com/accounts/${provider-account-name}`
	`  owner: consumer`
	`  ....`
	
	Note
	• In the example the subscriber subaccount is consuming a Solution that is provided by you.
	• The consumer property of the destination indicates that this destination is going to be deployed into the subaccount of the consumer.

`${landscape-url}`	Instructs the SAP BTP to resolve the placeholder value to the current landscape URL when deploying the destination. This placeholder can be used only in the `url` property for a destination, the `token-service-url` parameter, and in the `application-url` property that serves as a `provided` dependency of the `com.sap.java` and `java.tomcat` module types.
	
	Example
	`modules:`
	`  - name: java-module`
	`  type: com.sap.java`
	`  provides:`
	`  - name: java-module`
	`  properties:`
	`  application-url: myjava.${landscape-url}/`
	`  parameters:`
	`  name: exampleapp`
	`  ...`
	`  - name: abc-java`
	`  type: com.sap.java`
	`  parameters:`
	`  destinations:`
	`  - name: ExampleWebsite`
	`  type: HTTP`
	`  url: abc.${landscape-url}/`
	`  ....`

4.4.7 Integration with Transport Management Tools

To transport an application or application content to other subaccounts, you use the Enhanced Change and Transport System (CTS+) or the cloud-based Transport Management Service.

- Transport Management using the Enhanced Change and Transport System (CTS+)
  Use this option if you already have CTS+ in use for other applications, or if you have a hybrid landscape in which you want the ABAP system to be leading the transport environment.

To learn more about	See
How to use CTS+ to transport SAP Business Technology Platform applications from one subaccount to another	Transporting Multitarget Applications with CTS+ [page 1056]
What you need to do to enable the direct upload of MTA archives to a CTS+ transport request	Set Up a Direct Upload of MTA Archives to a CTS+ Transport Request [page 1058]
How to configure destinations to the target end points of deployment provided by the Solutions Lifecycle Management Service that are required as part of the setup of your transport landscapes in CTS+ and Transport Management Service.	Configuring the Access to the SAP Solution Lifecycle Management service [page 1061]

- Transport Management Service
  Use this option if you do not have any ABAP footprint, and if you have a cloud-based landscape.

To learn more about	See
How to use the Transport Management Service (BETA) in general	Introduction to the Transport Management Service
What you need to do to enable the direct upload of MTA Archives to a transport request that will be used by the Transport Management Service (BETA)	Set Up Direct Uploads of MTA Archives Using the Transport Management Service [page 1060]
How to configure destinations to the target end points of deployment provided by the Solutions Lifecycle Management Service that are required as part of the setup of your transport landscapes in CTS+ and Transport Management Service.	Configuring the Access to the SAP Solution Lifecycle Management service [page 1061]

4.4.7.1 Transporting Multitarget Applications with CTS+

You can enable transport of SAP BTP applications and application content that is available as Multitarget Applications (MTA) using the Enhanced Change and Transport System (CTS+).

Prerequisites

- You have configured your SAP Business Technology Platform subaccounts for transport with CTS+ as described in How To... Configure SAP BTP for CTS

- The content that you want to transport can be made available as a Multitarget Application (MTA) archive as described in Multitarget Applications for the Neo Environment [page 1016].

Context

You use the Change and Transport System (CTS) of ABAP to transport and deploy your applications running on SAP Business Technology Platform in the form of MTAs, for example, from development to a test or production subaccount. Proceed as follows to be able to transport an SAP BTP application:

Procedure

1. Package the application in a Multitarget Application (MTA) archive. To do this, you have the following options:
    1. Use the Cloud MTA Build Tool .
    2. Use the Solution Export Wizard as described in Exporting Solutions [page 1101].
2. Attach the MTA archive to a CTS+ transport request as described in How To... Configure SAP BTP for CTS.

    If you use the Solution Export Wizard to package the MTA archive, you can configure it for direct export to CTS+ . This means that a CTS+ transport request is automatically created when the MTA is exported and the archive file is attached to the transport request. The transport request is released and put in the import queue of the follow-on subaccount. Additional configuration steps are necessary for this and are described in Set Up a Direct Upload of MTA Archives to a CTS+ Transport Request [page 1058].
3. Trigger the import of an SAP BTP application as described in How To... Configure SAP BTP for CTS.

Related Information

Resources on CTS+

Setting up a CTS+ enabled transport landscape in SAP Business Technology Platform

4.4.7.2 Set Up a Direct Upload of MTA Archives to a CTS+ Transport Request

Use the CTS+ Export Web Service to perform a transport of a multitarget application from one subaccount to another.

Prerequisites

- You have activated and configured the CTS+ Export Web Service as described in Activating and Configuring CTS Export Web Service.

  > **i Note**
  >
  > Make sure that you select the *Transport Channel Authentication and User ID / Password* as a provider security of the web service binding.

  Note the *Calculated Access URL* of the web service, which can be found in the transport settings. The *Calculated Access URL* follows the pattern `/sap/bc/srt/rfc/sap/export_cts_ws/<ABAP Client ID>/export_cts_ws/export_cts_ws`.
- You have to define a user that is going to call the CTS+ Export Web Service. This user needs to have the following user roles:
  - `SAP_BC_WEBSERVICE_CONSUMER`
  - `SAP_CTS_PLUS`

  > **i Note**
  >
  > For more information about the CTS+ authorizations, see 1003674.
- You have installed and configured the Cloud Connector, which is used to connect on-premise systems with the SAP BTP. For more information, see Cloud Connector.
- You have exposed the CTS+ Export Web Service URLs as described in Configure Access Control (HTTP).

  > **i Note**
  >
  > If you maintain a list of trusted applications and a principal propagation trust configuration, you have to authorize the application `services:slservice`.
- Define the transport systems and route corresponding to your SAP BTP subaccounts. For more information, see How To... Configure HCP for CTS

Context

Procedure

1. In the SAP BTP cockpit, navigate to *Services* and enable the SAP BTP *Solution Lifecycle Management* service.
2. Define the destinations leading to the on-premise systems. Navigate to *Services* > *Solution Lifecycle Management* > *Configure Destinations* > *New Destination*.
3. For the new destination configuration, enter the required parameters:
   - Name: `TransportSystemCTS`

     > **Note**
     >
     > This destination name is mandatory.

   - Type: `HTTP`
   - URL: `<Exposed URL of the system, taken from the Cloud Connector section, following the convention: https://<virtual host name>:<virtual port, such as 443>/<Calculated Access URL>><System ID of the source system in the transport route, which is defined above>`

     > **Example**
     >
     > `https://myctsplushost:443/sap/bc/srt/rfc/sap/export_cts_ws/001/export_cts_ws/export_cts_ws`

   - Proxy Type: `OnPremise`
   - Authentication: `BasicAuthentication`
   - User: `<user name for the on-premise system, which is defined above>`
   - Password: `<password for the on-premise system user>`

   You also have the following additional properties:

   - (Mandatory) sourceSystemId: `<System ID of the source system in the transport route, which is defined above>`
   - (Optional) chunkSizeMB: `<desired chunk size for the transported MTA>`

   > **Note**
   >
   > You have to manually enter the attributes names, as they are not available in the drop-down list.

Results

For this feature to be consumed by the Cloud Platform Integration, see Content Transport.

Related Information

Transporting Multitarget Applications with CTS+ [page 1056]

4.4.7.3 Set Up Direct Uploads of MTA Archives Using the Transport Management Service

Create the required configurations for using the Transport Management Service to transport MTA archives between subaccounts.

Prerequisites

- You are subscribed and have access to the Transport Management Service, and have set up the environment to transport MTA archives directly in an application. For more information, see Set Up the Environment to Transport Content Archives Directly in an Application.
- You have a service key, which contains parameters you need to refer in the required destinations.

Context

To perform transports of MTA archives using the Transport Management Service, you have to create and set up destinations defining the source transport node for transporting MTA archives. Proceed as follows:

Procedure

1. In the SAP BTP cockpit, navigate to *Services* and enable the *Solution Lifecycle Management* service.
2. Navigate to the *Services* > *Solution Lifecycle Management* > *Configure Destinations*.
3. Choose *New Destination* to create the destination directed at the Transport Management Service URL and defining the source transport node. Enter the following parameters:
   - Name: `TransportManagementService`

     > **i Note**
     >
     > This destination name is mandatory.

   - Type: `HTTP`
   - URL: `<"uri" parameter specified in the Service Key>`
   - Authentication: `OAuth2ClientCredentials`
   - Proxy Type: `Internet`
   - Client ID: `<"clientid" specified in the Service Key>`
   - Client Secret: `<"clientsecret" specified in the Service Key>`
   - Token Service URL: `<"url" specified in the Service Key>`, followed by the path `/oauth/token`. For example:

> **Example**
>
> `https://tmsdemo123.authentication.sap.hana.ondemand.com/oauth/token`

- You also have the following additional properties:
  - (Mandatory) sourceSystemId: `<System ID of the source system in the transport route, which is defined above>`

> **Note**
>
> You have to manually enter the attributes names, as they are not available in the drop-down list.

4. Save the destination.

Results

You can use the Transport Management Service to transport MTA archives.

Related Information

Get Access
Set Up the Environment to Transport MTAs Directly in an Application
Creating Service Keys
Content Transport

4.4.7.4 Configuring the Access to the SAP Solution Lifecycle Management service

To deploy Multitarget Applications from other tools, such as CTS+ or the Transport Management Service, you have to connect to the SAP Solution Lifecycle Management service for SAP BTP by using its dedicated service endpoint **`https://slservice.<landscape-host>/slservice`**. You have two authentication methods available - Basic authentication, and OAuth Platform API Client.

> **Note**
>
> The default option for both CTS+ and Transport Management Service is Basic authentication.

The complete URL you have to use is based on the following patterns, respectively:

- **`https://slservice.<landscape-host>/slservice/slp/basic/<subaccount-technical-name>/slp`** - authentication using username and password
- **`https://slservice.<landscape-host>/slservice/slp/oauth/<subaccount-technical-name>/slp`** - authentication using an OAuth token created using the OAuth client

Landscape host can be found at Regions and Hosts Available for the Neo Environment [page 17].

Depending on your chosen method, proceed as follows:

- Basic authentication:
  1. Ensure the user has an assigned platform role that contains the following scopes:
     - *Manage Multitarget Applications*
     - *Read Multitarget Applications*

     For more information, see section Managing Member Authorizations in the Neo Environment [page 1301]
- Authentication using an OAuth Client:
  1. Create a new OAuth client as described in Using Platform APIs [page 1149].
  2. During the process, assign the following scopes from the SAP Solution Lifecycle Management service API:
     - *Manage Multitarget Applications*
     - *Read Multitarget Applications*

4.4.8 Modeling Solutions

In the context of SAP BTP, a solution is comprised of various application types and configurations, designed to serve a certain scenario or task flow. Typically the comprised parts of the solution are interconnected and have a common lifecycle. They are explicitly deployed, updated, deleted, configured, and monitored together.

A solution allows you to easily manage complex deployable artifacts. You can compose a solution by yourself, or you can acquire one from a third-party vendor. Furthermore, you can use the solutions to deploy artifacts that are comprised by entities external to SAP BTP, such as SAP SuccessFactors entities. This allows you to have a common management and lifecycle of artifacts spread across various SAP platforms and systems.

The following entities are required for creating a solution:

- A Multitarget Application (MTA) archive, which contains all required application types and configurations as well as a deployment descriptor file. It is intended to be used as a generic artifact that can be deployed and managed on several SAP BTP subaccounts. For example, you can reuse one MTA archive on your development and productive subaccounts.
- (Optionally) An MTA extension descriptor file that contains a deployment-specific data. It is intended to be used as a specific data source for a given SAP BTP subaccount. For example, you can have different extension descriptors for your development and productive subaccounts. Alternatively, you can also provide this data manually during the solution deployment.

You model the supported entities according to the MTA specification so that they can be deployed as a solution.

Related Information

Modeling Java Applications [page 1063]
Modeling Database Bindings [page 1066]
Modeling HTML5 Applications [page 1065]
Modeling Destinations [page 1072]

Modeling SAP SuccessFactors Extensions [page 1083]
Create a Hello World Multitarget Application [page 1017]

4.4.8.1 Modeling Java Applications

The SAP BTP allows you to deploy Java applications that run either on the proprietary SAP Java Web or the Java Web Tomcat runtime containers. These Java applications are `com.sap.java` and the `java.tomcat`.

When you model a Java application in the MTA deployment descriptor, you can specify a set of properties related to this application. For a complete list of the supported properties, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].

If a Java application is a part of your solution, the following rules apply during deployment:

- The Java application is deployed and started at the end of the deployment
- If a Java application with the same name already exists in your subaccount, it is replaced with the newer Java application
- An existing Java application is updated only if its binaries or configuration in the MTA deployment descriptor have been changed
- When updating an already existing application, parameters defined in the new MTA deployment descriptor override the existing parameters in the already deployed application. Parameters not defined in the descriptor are copied from the already existing application.
- You can also update a Java application using a rolling update. For more information, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].
- You can deploy one Java application that is distributed in two or more `war` files in the MTA archive. They have to be described accordingly in the `MANIFEST.MF` file, and the archive names should differ.

> **Note**
>
> Consider the following:
>
> - You have available resources to deploy a Java application to your subaccount
> - You have created the Java application `.war` archive
> - You have a database and valid credentials

The Java applications are modeled as a Multitarget Application (MTA) specification modules.

For specification of the Java application module, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].

For the examples below, we assume that you have the following:

- Database alias `tst`
- Database credentials `myuser` and `mypassword`

> **Example**
>
> **MTA Deployment Descriptor for com.sap.java**

```
_schema-version: '3.1'
parameters:
  hcp-deployer-version: '1.1.0'
```

```
ID: com.example.basic.javaapp
version: 0.1.0
modules:
  - name: example-java
    type: com.sap.java
    parameters:
       name: exampleapp
       jvm-arguments: -server
       java-version: JRE 7
       runtime-version: 1
    requires:
     - name: dbbinding
resources:
  - name: dbbinding
    type: com.sap.hcp.persistence
    parameters:
      id: tst
```

Example

MTA Deployment Descriptor for java.tomcat

```
_schema-version: '3.1'
parameters:
  hcp-deployer-version: '1.1.0'
ID: com.example.basic.javaapp
version: 0.1.0
modules:
  - name: example-java
    type: java.tomcat
    parameters:
       name: exampleapp
       jvm-arguments: -server
       java-version: JRE 8
       runtime-version: 3
    requires:
     - name: dbbinding
resources:
  - name: dbbinding
    type: com.sap.hcp.persistence
    parameters:
      id: tst
```

In the examples above you see the required application module properties such as the required Java version, runtime, description, and so on.

You also have to create an MTA extension descriptor that will hold sensitive data, such as credentials.

Note

Always enter the security-sensitive data of your Solution in an MTA extension descriptor.

Example

MTA Extension Descriptor

```
_schema-version: '3.1'
ID: com.example.basic.javaapp.config
extends: com.example.basic.javaapp
parameters:
  title: Java Application Example
  description: This is an example of the Java Application module
```

```
resources:
  - name: dbbinding
    parameters:
      user-id: myuser
      password : mypassword
```

In the example above, the extension descriptor adds the `user-id` and `password` parameters to the resource, which is modeled in the deployment descriptor.

After you deploy your solution, you can open its tile in the cockpit and check if the Java application is deployed.

Related Information

Operating Solutions [page 1089]
The Multitarget Application Model v.2
The Multitarget Application Model v.3

4.4.8.2 Modeling HTML5 Applications

You can deploy HTML5 applications to the SAP BTP by modeling it as a part of a Multitarget Application.

When you model an application in the MTA deployment descriptor, you have to specify a set of properties related to the application. For a complete list of the supported properties, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].

The following rules apply when you deploy a solution that contains an HTML5 application:

- If an application with an identical name but a different version already exists in your subaccount, the added new version exists in parallel to the earlier application. Depending on the value of the `active` parameter, the new version is activated.
- If an application with an identical name and the identical version already exists in your subaccount, the application in the solution to be deployed is going to be skipped.
- If there is no version specified in the MTA deployment descriptor, it will be deployed with its current timestamp as version.
- When you delete a solution containing an HTML5 application, the application itself and all of its versions are going to be deleted.

> **Note**
>
> The HTML5 application has to be packed in a ZIP archive.

> **Example**

> **Sample Code**

```
parameters:
  hcp-deployer-version: '1.1.0'
```

```
ID: com.sap.example.html5
version: 0.1.0
modules:
  - name: examplehtml5
    type: com.sap.hcp.html5
    parameters:
      name: example
      version: '0.1.0'
      active: true
      display-name: Example HTML5
```

To always create a new version of the HTML5 application, you can also use the `${timestamp}` as a suffix to you version.

Example

```
- name: examplehtml5
  type: com.sap.hcp.html5
  parameters:
    name: example1
    version: '0.1.0-${timestamp}'
```

Related Information

Operating Solutions [page 1089]
The Multitarget Application Model v.2
The Multitarget Application Model v.3

4.4.8.3 Modeling Database Bindings

By using a database binding, the Java applications connects to a database set up in your current subaccount or provided by another subaccount part of the same global account. This connection is modeled within your solution by setting it up during the deployment operation.

Note

Ensure the following prerequisites:

- You have a database that is set up in your subaccount or there is a database provided to you by another subaccount.
- You have valid credentials for that database. In case that you do not have valid credentials for the database, default credentials will be generated for you.

You can create a database binding to one of the following:

- The `<DEFAULT>` data source of your Java application
- A named data source of your Java application by using the parameter `binding-name` that is added to the database binding resource required in the `requires` section of the `com.sap.java` and `java.tomcat` module types.

For specification of the named database binding parameter, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].

> **Note**
>
> You cannot have a database binding to the `<DEFAULT>` data source together with a database binding to a named data source, but you can have more than one database binding to named data sources.

The following rules apply:

Does a Binding Already Exist?	Are User Credentials Provided?	Action
No	Yes	Creates a new database binding using the provided credentials
No	No	Creates a new database binding with generated credentials
Yes	Yes	Creates a new database binding using the provided credentials
Yes	No	Uses the old database binding

Each database binding is modeled as a Multitarget Application (MTA) resource, which is required by a Java application module.

For specification of the database binding resource, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].

First, you have to model the deployment descriptor that will contain the Java application module and the database binding resource and then you have to create an extension descriptor that will hold sensitive data, such as credentials.

> **Note**
>
> Make sure that you always use an extension descriptor when you have sensitive data within your solution.

Modeling a Database Binding to a Database within Your Subaccount

Database Binding to the `<DEFAULT>` Data Source

For the example below, we assume that you have the following:

- Database alias `tst`
- Database credentials `myuser` and `mypassword`

> **Example**
>
> **MTA Deployment Descriptor**

```
_schema-version: '3.1'
parameters:
  hcp-deployer-version: '1.1.0'
ID: com.example.basic.dbbinding
version: 0.1.0
```

```
modules:
  - name: example-java
    type: com.sap.java
    parameters:
       name: exampleapp
       jvm-arguments: -server
       java-version: JRE 7
       runtime: neo-java-web
       runtime-version: 1
    requires:
     - name: dbbinding
resources:
  - name: dbbinding
    type: com.sap.hcp.persistence
    parameters:
      id: tst
```

In the example above you can see the following:

- The Java application module requires the database binding resource
- The database binding resource does not specify any credentials

Example

MTA Extension Descriptor

```
_schema-version: '3.1'
ID: com.example.basic.dbbinding.config
extends: com.example.basic.dbbinding
parameters:
  title: Database binding example
  description: This is an example of the database binding resource
resources:
  - name: dbbinding
    parameters:
      user-id: myuser
      password : mypassword
```

In the example above, the extension descriptor adds the `user-id` and `password` parameters to the resource dbbinding, which is modeled in the deployment descriptor.

Named Database Bindings to Specified Data Sources

For the example below, we assume that you have the following:

- Database aliases `tst` and `abc`
- Database credentials `myuser` and `mypassword`

Example

MTA Deployment Descriptor

```
_schema-version: '3.1'
parameters:
  hcp-deployer-version: '1.1.0'
ID: com.example.basic.dbbinding
version: 0.1.0
modules:
  - name: example-java
    type: com.sap.java
    parameters:
       name: exampleapp
```

```
        jvm-arguments: -server
        java-version: JRE 7
        runtime: neo-java-web
        runtime-version: 1
    requires:
      - name: firstdbbinding
        parameters:
          binding-name: tstbinding
      - name: seconddbbinding
        parameters:
          binding-name: abcbinding
resources:
  - name: fisrtdbbinding
    type: com.sap.hcp.persistence
    parameters:
      id: tst
  - name: seconddbbinding
    type: com.sap.hcp.persistence
    parameters:
      id: abc
```

In the example above you can see the following:

- The Java application module requires the database binding resources
- The `binding-name` parameters are added to each database binding under the `requires` section
- The database binding resources do not specify any credentials

Example

MTA Extension Descriptor

```
_schema-version: '3.1'
ID: com.example.basic.dbbinding.config
extends: com.example.basic.dbbinding
parameters:
  title: Named Database bindings example
  description: This is an example of the database binding resources
resources:
  - name: firstdbbinding
    parameters:
      user-id: myuser
      password : mypassword
  - name: seconddbbinding
    parameters:
      user-id: myuser
      password : mypassword
```

In the example above, the extension descriptor adds the `user-id` and `password` parameters to each resource, which is modeled in the deployment descriptor.

Modeling a Database Binding to a Database that is Provided by Another Subaccount

Database Binding to the <DEFAULT> Data Source

For the example below, we assume that you have the following:

- Database alias `tst`, which is provided by another subaccount

  > **i Note**
  >
  > The provider subaccount must belong to the same global account to which your subaccount belongs.

  The provider subaccount is `abcd`
  Database credentials `myuser` and `mypassword`

> **Example**
>
> **MTA Deployment Descriptor**

```
_schema-version: '3.1'
parameters:
  hcp-deployer-version: '1.1.0'
ID: com.example.basic.dbbinding
version: 0.1.0
modules:
  - name: example-java
    type: com.sap.java
    parameters:
       name: exampleapp
       jvm-arguments: -server
       java-version: JRE 7
       runtime: neo-java-web
       runtime-version: 1
    requires:
     - name: dbbinding
resources:
  - name: dbbinding
    type: com.sap.hcp.persistence
    parameters:
      id: tst
      account: abcd
```

In the example above, you can see the following:

- The Java application module requires the database binding resource
- The database binding resource does not specify any credentials
- The provider subaccount `abcd` is specified as a resource parameter

> **Example**
>
> **MTA Extension Descriptor**

```
_schema-version: '3.1'
ID: com.example.basic.dbbinding.config
extends: com.example.basic.dbbinding
parameters:
  title: Database binding example
  description: This is an example of the database binding resource
resources:
```

```
  - name: dbbinding
    parameters:
      user-id: myuser
      password : mypassword
```

In the example above, the MTA extension descriptor adds the `user-id` and `password` parameters to the resource `dbbinding`, which is modeled in the MTA deployment descriptor. After you deploy your solution, you can open its tile in the cockpit and check if the database binding is created.

Named Database Bindings to Specified Data Sources

For the example below, we assume that you have the following:

- Database aliases `tst` and `abc`, which are provided by another subaccount

  > **i Note**
  >
  > The provider subaccount must belong to the same global account to which your subaccount belongs.

  The provider subaccounts are `abcd` and `test`
  Database credentials `myuser` and `mypassword`

> **Example**
>
> **MTA Deployment Descriptor**

```
_schema-version: '3.1'
parameters:
  hcp-deployer-version: '1.1.0'
ID: com.example.basic.dbbinding
version: 0.1.0
modules:
  - name: example-java
    type: com.sap.java
    parameters:
       name: exampleapp
       jvm-arguments: -server
       java-version: JRE 7
       runtime: neo-java-web
       runtime-version: 1
    requires:
      - name: firstdbbinding
        parameters:
          binding-name: tstbinding
      - name: seconddbbinding
        parameters:
          binding-name: abcbinding
resources:
  - name: firstdbbinding
    type: com.sap.hcp.persistence
    parameters:
      id: tst
      account: abcd
  - name: seconddbbinding
    type: com.sap.hcp.persistence
    parameters:
      id: abc
      account: test
```

In the example above, you can see the following:

- The Java application module requires the database binding resources
- The `binding-name` parameters are added to each database binding under the `requires` section
- The database binding resources do not specify any credentials
- The provider subaccounts `abcd` and `test` are specified as resource parameters

> ### Example
>
> **MTA Extension Descriptor**
>
> ```
> _schema-version: '3.1'
> ID: com.example.basic.dbbinding.config
> extends: com.example.basic.dbbinding
> parameters:
>   title: Named database bindings example
>   description: This is an example of the database binding resources
> resources:
>   - name: firstdbbinding
>     parameters:
>       user-id: myuser
>       password : mypassword
>   - name: seconddbbinding
>     parameters:
>       user-id: myuser
>       password : mypassword
> ```

In the example above, the MTA extension descriptor adds the `user-id` and `password` parameters to each resource, which is modeled in the MTA deployment descriptor. After you deploy your solution, you can open its tile in the cockpit and check if the database bindings are created.

> ### Note
>
> When you delete a database binding, the credentials that you used are not removed from the database. Delete them manually, if you want to do so.

Related Information

Operating Solutions [page 1089]
The Multitarget Application Model v.2
The Multitarget Application Model v.3

4.4.8.4 Modeling Destinations

You can connect your applications to another source by describing the source connection properties in a destination. Later on, you can access that destination from your application.

Depending on whether the destination source is located within the SAP BTP or not, the destinations are classified as internal or external. You can also provide a Solution for consumption to another SAP BTP subaccount and define a destination as deployable to all subscriber subaccounts.

The supported destination levels you can model within a Solution are:

Supported Levels of Destinations

Level	Description	Support for Java Applications	Support for Tomcat Applications	Support for HTML5 Applications
Subaccount-Level Destination	After deployment, it is available for all applications within your subaccount	Yes	Yes	Yes
Application-Level Destination	After deployment, it is available only for designated applications	Yes	Yes	No

Related Information

Modeling Subaccount-Level Destinations [page 1073]
Modeling Application-Level Destinations [page 1078]

4.4.8.4.1 Modeling Subaccount-Level Destinations

Subaccount-level destinations are not linked to a particular application, but instead can be used by all applications. For example, the subaccount-level destination can be used by an HTML5 application to connect to a source Java application.

> **i Note**
>
> If you modify a subaccount-level destination, you will affect all applications that use it. The subaccount-level destination has a lifecycle that is independent from the applications that use it.

Destinations to External Resources

Destinations to external resources lead to services or applications that are not part of the current Multitarget Application (MTA) archive and you do not have direct access to them. For example, it might be an application running in another subaccount or outside SAP BTP.

When you want to describe subaccount-level destinations to external resources, the modeling is as a module of type `com.sap.hcp.destination`. In this type of destination relations, first you declare that a module requires the dependency using a `requires` element, and then you provide the dependency details as module type parameters. The subaccount-level destination has a lifecycle that is independent from the applications that use it. Note that if you need your Java application to have more than one destination, you have to model each subaccount-level destination in a separate module.

For a list of the available destination parameters, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027], and the design documents The Multitarget Application Model v.2 and The Multitarget Application Model v.3.

→ Remember

Note the following important cases when it comes to deploying a destination:

- If you need more than one destination, you have to model each subaccount-level destination in a separate module.
- Use the property `force-overwrite` to choose a redeployment approach for an existing destination. Place the property at destination level, and use the value `true` to have the destination forcefully overwritten, or `false` to leave it unchanged, which is also the default behavior.
- If you want a destination to be created before the application that requires it, use a `required` dependency to instruct SAP BTP to define the deploy order. For example, if you have a Java application with a destination defined in its `web.xml` file, you have to use the `required` dependency so that the destination is resolved correctly when the Java application starts.

The following shows how to create a subaccount-level destination:

Example

```
modules:
  - name: nwl
    type: com.sap.java
    requires:
      - name: examplewebsite-connection
    parameters:
      name: networkinglunch
      ...

  - name: examplewebsite-connection
    type: com.sap.hcp.destination
    parameters:
      name: ExampleWebsite
      type: HTTP
      description: Connection to ExampleWebsite
      url: http://www.examplewebsite.com
      proxy-type: Internet
      authentication: BasicAuthentication
      user: myuser
      password: mypassword
   ...
```

In the example above, the module type `com.sap.hcp.destination` is used to define the subaccount-level destination and the Java module `nwl` requires it, because the destination is created prior starting the Java application. The required section has to ensure proper ordering.

The example above will result in a subaccount-level destination created within the consumer subaccount with credentials that are still placed into the MTA extension descriptor. If you are providing your solution for consumption by another subaccount, you might want to create that destination into the subscriber subaccount. To do this, you have to use the `owner` option.

Example

MTA Deployment Descriptor

```
_schema-version: '3.1'
```

```
parameters:
  hcp-deployer-version: '1.1.0'
ID: com.example.basic.destination.subaccount
version: 0.1.0
modules:
  - name: nwl
    type: com.sap.java
    requires:
      - name: examplewebsite-connection
    parameters:
      name: networkinglunch
  - name: examplewebsite-connection
    type: com.sap.hcp.destination
    requires:
      - name: data-storage
    parameters:
      name: ExampleWebsite
      type: HTTP
      description: Connection to ExampleWebsite
      url: http://www.examplewebsite.com
      proxy-type: Internet
      authentication: BasicAuthentication
      user: ~{data-storage/user}
      password: ~{data-storage/password}
      owner: consumer
resources:
  - name: data-storage
    properties:
      user:
      password:
```

MTA Extension Descriptor

```
_schema-version: '3.1'
parameters:
  hcp-deployer-version: '1.1.0'
ID: com.example.basic.destination.subaccount.config
extends: com.example.basic.destination.subaccount
version: 0.1.0
modules:
resources:
  - name: data-storage
    properties:
      user: myuser
      password: mypassword
```

In the example above, the `examplewebsite-connection` destination is deployed to the subscriber subaccount. The untyped resource `data-storage` contains the sensitive parameters, which are deployed using the MTA extension descriptor.

> **i Note**
>
> - The reference syntax `${source/value}` is used to link the destinations `user` and `password` options.
> - The `data-storage` resource is of `untyped` type.
>
> For more information, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027]

In the example above, you create the destination within the subscriber subaccount, but the credentials for that destination are still provided by you. If the consumer of your solution has to provide the credentials for the destination, you have to use the `consumer-optional` metadata element.

Note

Note that using metadata is available in an MTA archive with schema version `3.1` and higher.

Example

MTA Deployment Descriptor

```
_schema-version: '3.1'
parameters:
  hcp-deployer-version: '1.1.0'
ID: com.example.basic.destination.subaccount
version: 0.1.0
modules:
  - name: nwl
    type: com.sap.java
    requires:
      - name: examplewebsite-connection
    parameters:
      name: networkinglunch
  - name: examplewebsite-connection
    type: com.sap.hcp.destination
    requires:
      - name: data-storage
    parameters:
      name: ExampleWebsite
      type: HTTP
      description: Connection to ExampleWebsite
      url: http://www.examplewebsite.com
      proxy-type: Internet
      authentication: BasicAuthentication
      user: ~{data-storage/user}
      password: ~{data-storage/password}
      owner: consumer
resources:
  - name: data-storage
    properties:
      user:
      password:
    properties-metadata:
      user:
        optional: true
        consumer-optional: false
      password:
        optional: true
        consumer-optional: false
```

MTA Extension Descriptor of the Provider

```
_schema-version: '3.1'
ID: com.example.basic.destination.subaccount.config
extends: com.example.basic.destination.subaccount
parameters:
  title: Subaccount Destination Example
  description: This is an example of the sample Subaccount Destination
```

MTA Extension Descriptor of the Subscriber

```
_schema-version: '3.1'
ID: com.example.basic.destination.subaccount.config.subscriber
extends: com.example.basic.destination.subaccount.config
resources:
  - name: data-storage
    properties:
```

```
      user: subscriberuser
      password: subscriberpassword
```

In the example above, the `consumer-optional` metadata is used to enforce the subscriber to provide credentials. The provider's extension descriptor does not provide the credentials, but the consumer's one.

> **i Note**
>
> - The `consumer-optional` metadata is used together with the `optional` option.
> - If you do not use the `consumer-optional` metadata when you deploy the solution to your subaccount, the operation will fail due to missing data.

Destinations to Internal Applications

The subaccount-level destination to an internal application is a destination of type `HTTP` that points to a Java application, which in turn is part of the current MTA and will be deployed with the same solution. It is modeled as a `com.sap.hcp.destination` module type.

> **i Note**
>
> - If you need more than one destination, you have to model each subaccount-level destination in a separate module.
> - To overwrite an already existing destination, you have to use the `force-overwrite` option. For more information, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].

In the following example, an HTML5 application, which uses a subaccount-level destination to an internal resource, connects to a Java application as a backend. The destination uses the URL of the Java application as its URL target:

> **Example**

```
- name: abc
  type: com.sap.java
  provides:
    - name: abc
      properties:
        application-url: ${default-url}
  parameters:
    name: networkinglunch
    ...

- name: abc-destination
  type: com.sap.hcp.destination
  requires:
    - name: abc
  parameters:
    name: NetworkingLunchBackend
    type: HTTP
    url: ~{abc/application-url}
    proxy-type: Internet
    authentication: AppToAppSSO

- name: abc-ui
```

```
  type: com.sap.hcp.html5
  requires:
    - name: abc-destination
  parameters:
    name: networkingui
```

In the example above, you can see the following:

- The Java module type `abcapplication-url` property, whose value is a placeholder. For more information about the `${default-url}` placeholder, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].
- The destination `abc-destination` is a module of type `com.sap.hcp.destination` and requires the Java application module.
  By requiring the Java application module, the destination can access all provided properties of that module (namely the `application-url` property). Later on, the destination module can use a reference to read the provided properties.
- The destination module uses the reference to read the value of the `application-url` property.
- The HTML5 module requires the destination module, because the destination has to be created prior to deploying the HTML5 application. The required section will ensure the proper ordering.
  For more information about the relation between HTML5 applications and destinations, see Assign Destinations for HTML5 Applications [page 1636].

> **i Note**
>
> Ensure that your applications do not have circular dependencies, that is, that the `application-url` provides its property or one Java application does not refer to the `application-url` property of another Java application module.

4.4.8.4.2 Modeling Application-Level Destinations

Application-level destinations apply only within a given application, compared to subaccount-level destinations that apply to the whole subaccount. You can use them to connect your application to resources outside the SAP BTP, applications not part of your subaccount, applications from your subaccount and even to your own application.

Destinations to External Resources

Destinations to external resources lead to services or applications external to and not accessible by the current Multitarget Application (MTA) archive. For example, it can be an application running in another subaccount.

Application-level destinations to external resources are modeled as items within the `destinations` parameter of the `com.sap.java` module type. This means that the lifecycle of such a destination is bound to the lifecycle of the corresponding application.

For a list of the available destination parameters, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027], and the design documents The Multitarget Application Model v.2 and The Multitarget Application Model v.3.

→ Remember

Note the following special cases when it comes to deploying a destination:

- If you need more than one destination, you have to model each subaccount-level destination in a separate module.
- Use the property `force-overwrite` to choose a redeployment approach for an existing destination. Place the property at destination level, and use the value `true` to have the destination forcefully overwritten, or `false` to leave it unchanged, which is also the default behavior.
- You cannot define a destination in the `web.xml` file of the Java application, if that specific destination points to the application itself, and has a URL automatically resolved by the SAP BTP. You have to manually resolve the destination in the application code.

The following shows how to create an application-level destination to an external resource:

Example

```
modules:
  - name: abc
    type: com.sap.java
    parameters:
      name: networking
      ...
      destinations:
        - name: ExampleHttpDestination
          type: HTTP
          url: http://www.examplewebsite.com
          proxy-type: Internet
          authentication: BasicAuthentication
          user: myuser
          password: mypassword
```

In the example above, you can see the following:

- The `com.sap.java` module defines the Java application that has to be deployed.
- The `destinations` parameter defines the destinations that have to be created.

As a result, the example above creates an application-level destination within your subaccount with credentials, which are still located in the MTA deployment descriptor.

If you want to provide your solution for consumption by another subaccount, you can create that destination into the subscriber subaccount. To do this, you can use the `owner` option.

Example

Deployment Descriptor

```
modules:
  - name: abc
    type: com.sap.java
    parameters:
      name: networking
      requires:
        - name: data-storage
      ...
      destinations:
        - name: ExampleDestination
          type: HTTP
          url: http://www.examplewebsite.com
```

```
          proxy-type: Internet
          authentication: BasicAuthentication
          user: ~{data-storage/user}
          password: ~{data-storage/password}
          owner: consumer
      ...
resources:
 - name: data-storage
   properties:
     user:
     password:
     ...
```

Extension Descriptor

```
...
resources:
  - name: data-storage
    properties:
      user: myuser
      password : mypassword
      ...
```

In the example above, you can see the following:

- The `ExampleDestination` destination will be deployed to the subscriber subaccount.

  > **i Note**
  >
  > The `owner` option indicates that the destination has to be deployed to the subscriber subaccount.

- The untyped resource `data-storage` contains the sensitive parameters, which are deployed using the MTA extension descriptor.

  > **i Note**
  >
  > Be aware of the following:
  >
  > - The reference syntax `~{source/value}` is used to link the destinations `user` and `password` options.
  > - The `data-storage` resource is of `untyped` type.
  >
  > For more information, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027]

The example above creates the destination within the subscribers subaccount, but the credentials for that destination are still provided separately by you. If the consumer of your solution has to provide the credentials for the destination, you have to use `consumer-optional` metadata.

> **i Note**
>
> Note that metadata is available in an MTA archive with schema version `3.1` and higher.

> **Example**
>
> **Deployment Descriptor**

```
modules:
  - name: abc
```

```
    type: com.sap.java
    parameters:
      name: networking
      requires:
        - name: data-storage
      ...
      destinations:
        - name: ExampleDestination
          type: HTTP
          url: http://www.examplewebsite.com
          proxy-type: Internet
          authentication: BasicAuthentication
          user: ~{data-storage/user}
          password: ~{data-storage/password}
          owner: consumer
      ...

resources:
  - name: data-storage
    properties:
      user:
      password:
    properties-metadata:
      user:
        optional: true
        consumer-optional: false
      password:
        optional: true
        consumer-optional: false
    ...
```

Provider Extension Descriptor

```
...
# no credentials provided
...
```

Subscriber Extension Descriptor

```
resources:
  - name: data-storage
    properties:
      user: subscriberuser
      password: subscriberpassword
    ...
```

In the example above the `consumer-optional` metadata is used to enforce the consumer of your solution to provide the required credentials. In this case, the MTA extension descriptor of the consumer provides the required credentials instead of the provider MTA extension descriptor.

> **i Note**
>
> The `consumer-optional` metadata is used together with the `optional` option.
>
> If you do not use the `consumer-optional` metadata when you deploy the solution to your subaccount, the operation will fail due to missing data.

Destinations to Internal Applications

The application-level destination to an internal application is an `HTTP` type destination, which can point to the same or different Java application deployed with the same solution. It is modeled as a `com.sap.java` or `java.tomcat` module.

> **i Note**
>
> - If you need more than one destination, you have to model each subaccount-level destination in a separate module.
> - To overwrite an already existing destination, you have to use the `force-overwrite` option. For more information, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].

The following example creates a frontend Java application that has a destination directed at a backend Java application, which is deployed within the same solution. In this case, the URL of the source Java application is not described, but instead left to be resolved to its default value during deployment by the SAP BTP.

For additional options of what can be resolved automatically by the SAP BTP during deployment see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].

> **Example**

```
- name: abc
  type: com.sap.java
  provides:
    - name: abc
      properties:
        application-url: ${default-url}
  parameters:
    name: javaapp1
    ...
- name: abc-ui
  type: com.sap.java
  requires:
    - name: abc
  parameters:
    name: javaapp2
    ...
    destinations:
      - name: JavaAppBackend
        type: HTTP
        url: ~{abc/application-url}
        proxy-type: Internet
        authentication: AppToAppSSO
```

In the example above, you can see the following:

- The Java `abc` module provides the URL as an `application-url` property.

  > **i Note**
  >
  > The naming of the `applicaiton-url` property is mandatory. For more details, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].

- The value of the `application-url` property is a placeholder. For more details about the `${default-url}` placeholder, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027]

- The destination `JavaAppBackend` is an entry of the destinations parameter of the `com.sap.java` module.
- The module `abc-ui` requires the module abc. By requiring the `abc` module, the `abc-ui` module gains access to all provided properties of that module, namely the `application-url` property. Later on, the destination module can use a reference to read the provided properties. For more details, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].
- The destination is using the reference to read the value of the `application-url` property.

> **i Note**
>
> Ensure that your applications do not have circular dependencies, that is, that the `application-url` property or one Java Application does not refer to the `application-url` property of another Java Application module.

4.4.8.5 Modeling SAP SuccessFactors Extensions

You can connect your SAP SuccessFactors system to your SAP Business Technology Platform(SAP BTP) subaccount. After you do so, you can define a solution that extends it. In more complex scenarios, you can even provide a solution that can be consumed by another SAP BTP subaccount and extend the subscribers SAP SuccessFactors system.

> **i Note**
>
> Make sure that:
>
> - You have onboarded an SAP SuccessFactors company in your SAP BTP subaccount. If you are providing a solution that is consumed by another subaccount in the SAP BTP, the subscriber subaccount is responsible for onboarding the SAP SuccessFactors company. See Configuring the Subaccount in SAP BTP for SAP SuccessFactors [page 1177].
> - You have a database and valid credentials.

In the example below, you will create a standard SAP SuccessFactors extension. The “Benefits” sample Java application provided by SAP is used. It is located at https://github.com/SAP/cloud-sfsf-benefits-ext .

> **i Note**
>
> If you intend to provide this solution for consumption by other subaccounts:
>
> - The sample “Benefits” Java Application will be deployed to your subaccount, but the SAP SuccessFactors artifacts will be deployed to the subscriber subaccounts and their SAP SuccessFactors systems.
> - You can define an additional MTA extension descriptor for your subscribers, so that they can add their own specific data.

As a result, their solution is going to have:

- The sample “Benefits” Java application
- A database binding for the sample “Benefits” Java application
- The SAP SuccessFactorstile and SAP SuccessFactors role are going to be deployed to the SAP SuccessFactors company

- Connectivity for the sample “Benefits” Java application to an SAP SuccessFactors system

> **i Note**
>
> For the example below, we assume that you have the following:
>
> - Database alias `tst`
> - Database credentials `myuser` and `mypassword`

You have to model the sample “Benefits” Java application as a module into the MTA deployment descriptor. You also have to define an SAP SuccessFactors Role module and a database binding resource.

> **Example**

```
_schema-version: '3.1'
parameters:
  hcp-deployer-version: '1.1.0'
ID: com.example.basic.sfsf
version: 0.1.0
modules:
  - name: benefits-app
    type: com.sap.java
    parameters:
       name: benefits
       jvm-arguments: -server
       java-version: JRE 7
       runtime: neo-java-web
       runtime-version: 1
       sfsf-idp-access: true
       sfsf-connections:
         - type: default
           additional-properties:
              nameIdFormat: 'urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified'
         - type: technical-user
           technical-user-id: SFSFAdmin
       sfsf-outbound-connections:
         - type: OAuth2SAMLBearerAssertion
           name: BenefitsOutboundConnection
           subject-name-id: mail
           subject-name-id-format: EMAIL_ADDRESS
           assertion-attributes-mapping:
              firstname: firstname
              lastname: lastname
              email: email
       role-provider: sfsf
       sfsf-home-page-tiles:
         resource: resources/benefits-tiles.json
    requires:
     - name: dbbinding
     - name: benefits-roles
  - name: benefits-roles
    type: com.sap.hcp.sfsf-roles
resources:
  - name: dbbinding
    type: com.sap.hcp.persistence
    parameters:
      id: tst
```

In the example above, you can see the following:

- The sample “Benefits” Java application module requires both database binding resource and the SAP SuccessFactors Roles module.

- The SAP SuccessFactors tile is defined as a parameter of the sample “Benefits” Java application and points to a `JSON` file within the Multitarget Application archive
- SAP SuccessFactors provider is defined as a parameter of the sample “Benefits” Java application
- The sample “Benefits” Java application is going to use the SAP SuccessFactors IDP for authentication
- The sample “Benefits” Java application is going to use the default connectivity options when accessing the SAP SuccessFactors system

Both the SAP SuccessFactors roles and tiles require additional files to be added to the Multitarget Application archive. The deployment descriptor contains only the modeling of those entities, but their actual content is external to the MTA deployment descriptor, in the same way as the sample “Benefits” Java application `.war` archive.

You also have to create a `JSON` file `benefits-tiles.json` that contains the SAP SuccessFactors tiles.

Example

```
[
    {
        "name" : "SAP Corporate Benefits",
        "path" : "com.sap.hana.cloud.samples.benefits",
        "size" : 3,
        "padding" : false,
        "roles" : ["Corporate Benefits Admin"],
        "metadata" : [
            {
                "title" : "SAP Corporate Benefits",
                "description" : "SAP Corporate Benefits home page tile",
                "locale" : "en_US"
            }
        ]
    }
]
```

In the example above, you can see an example of an SAP SuccessFactors tile for the sample “Benefits” Java application.

Next you have to create a `JSON` file `benefits-roles.json` that contains the SAP SuccessFactors roles.

Example

```
[
    {
        "roleDesc": "SAP Corporate Benefits Administrator",
        "roleName": "Corporate Benefits Admin",
        "permissions": []
    }
]
```

In the example above, you can see an example of an SAP SuccessFactors role for the sample “Benefits” Java application.

Afterward, you have to create your `MANIFEST.MF` file and define the Java application, roles, and tiles.

Example

```
Manifest-Version: 1.0
Created-By: SAP SE
Name: resources/benefits-roles.json
```

```
Content-Type: application/json
MTA-Module: benefits-roles
Name: com.sap.hana.cloud.samples.benefits.war
Content-Type: application/zip
MTA-Module: benefits-app
```

In the example above, you can see the following:

- Entry that links your SAP SuccessFactors roles with the MTA deployment descriptor
- Entry that links your “Benefits” sample Java application with the MTA deployment descriptor

Now you can create your Multitarget Application archive by following the JAR file specification. The archive structure has to be as follows:

Example

```
/com.sap.hana.cloud.samples.benefits.war
/META-INF
/META-INF/mtad.yaml
/META-INF/MANIFEST.MF
/resources/benefits-roles.json
/resources/benefits-tiles.json
```

Start by creating an MTA extension descriptor that holds the security-sensitive data, such as credentials.

Note

Make sure that you always use an extension descriptor when you have sensitive data within your solution.

Example

```
_schema-version: '3.1'
ID: com.example.basic.sfsf.config
extends: com.example.basic.sfsf
parameters:
  title: SuccessFactors example
  description: This is an example of the sample Benefits Java Application for
SuccessFactors
resources:
  - name: dbbinding
    parameters:
      user-id: myuser
      password : mypassword
```

In the example above, the extension descriptor adds the `user-id` and `password` parameters to the resource dbbinding, which is modeled in the deployment descriptor.

After you deploy your solution, you can open its tile in the cockpit and check if the SAP SuccessFactors extension solution is deployed.

Related Information

Operating Solutions [page 1089]
The Multitarget Application Model v.2

The Multitarget Application Model v.3
JAR file specification

4.4.8.6 Modeling Security Groups and Role Assignments

To organize application security roles and to manage user access, you create authorization groups in SAP BTP.

You model security groups in the MTA deployment descriptor using the module type `com.sap.hcp.group`. You can also assign any roles defined in a Java application to these authorization groups.

The following rules apply when you deploy a solution containing authorization groups:

- If the group already exists, it is updated with the new roles assignment defined in the MTA deployment descriptor.
- If you delete a solution, a group is not deleted, as it might be used by other applications.

Example

We assume that you have defined as follows a set of security roles in the `web.xml` of your Java application.

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>My Java Web Application</display-name>
  <security-role>
    <role-name>administrator</role-name>
  </security-role>
</web-app>
```

For a complete list of the supported properties, see MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment [page 1027].

The security roles can be assigned to a group modeled in the MTA deployment descriptor.

Example

```
ID: com.sap.mta.demo
_schema-version: '2.1'
parameters:
  hcp-deployer-version: '1.1.0'
modules:
- name: administratorGroup
   parameters:
      name: &adminGroup AdministratorGroup
   type: com.sap.hcp.group
- name: demowebapp
   parameters:
      name: demowebapp
      title: Demo MTA Application
      runtime-version: '3'
      java-version: JRE 8
      roles:
       - name: administrator
         groups:
          - *adminGroup
   requires:
```

```
    - name: administratorGroup
```

When you deploy the above example, a new authorization group named “AdministratorGroup” is created, and the “administrator” application security role form the “demowebapp” is assigned to this group. In case the roles already exists, only the application security role is assigned to the existing group.

Related Information

Role Assignment of HTML 5 Roles to Security Groups [page 1089]
Role Assignment of Fiori Roles to Security Groups [page 1088]
Operating Solutions [page 1089]
The Multitarget Application Model v.2
The Multitarget Application Model v.3

4.4.8.6.1 Role Assignment of Fiori Roles to Security Groups

You can assign security roles on subscription level for use with SAP Fiori applications.

These roles are assigned to authorization groups when designed as modules in a descriptor file, as shown in the following example:

Sample Code

```
ID: com.sap.mta.demo
_schema-version: '3.1'
modules:
 - name: administratorGroup
   parameters:
      name: &adminGroup AdministratorGroup
   type: com.sap.hcp.group
 - name: fiori-role
   type: com.sap.fiori.role
   parameters:
     name: HRManager
     groups:
          - *adminGroup
```

4.4.8.6.2 Role Assignment of HTML 5 Roles to Security Groups

You can assign security roles on subscription level for use with HTML5 applications.

These roles are assigned to authorization groups when designed as modules in a descriptor file, as shown in the following example:

Sample Code

```
ID: com.sap.mta.demo
_schema-version: '3.1'
modules:
 - name: administratorGroup
   parameters:
      name: &adminGroup AdministratorGroup
   type: com.sap.hcp.group
 - name: html5-role
   type: com.sap.hcp.html5.role
   parameters:
     name: HRManager
     groups:
          - *adminGroup
     requires:
          - name: administratorGroup
```

Related Information

Modeling HTML5 Applications [page 1065]
Modeling Security Groups and Role Assignments [page 1087]

4.4.9 Operating Solutions

You can deploy, update, monitor, and delete a solution.

To operate a solution you require at least one of the following roles in your subaccount:

- Default role *Administrator*
- Default role *Developer*
- A custom role with all following permissions: `manageMultiTargetApplication`, `readMultiTargetApplication`, `readJavaApplications`, and `manageJavaApplications`. For more information about custom roles, see Managing Roles [page 1710] and Manage Custom Platform Roles [page 1306].

> **Note**
>
> Currently you can operate SAP SuccessFactors extensions using only the *Administrator* or *Developer* roles.

Deploying Solutions

Depending on the type of the solution, you can operate it using the cockpit, CTS+ and the SAP Business Technology Platform(SAP BTP) console client for the Neo environment:

Standard Solution

This solution is only deployed and can be used in the current SAP BTP subaccount and subscription to it is not possible. All entities that are part of the solution will be deployed and managed within this subaccount.

You can deploy a standard solution using the:

- Cockpit: Deploy a Standard Solution [page 1093]
  By using the cockpit, you can deploy all of the solution artifacts to your subaccount in one go.
- Transport Managament Service: Integration with Transport Management Tools [page 1056]
  You can use the CTS+ to transport and promote a solution among different environments or landscapes.

  > **i Note**
  >
  > The CTS+ cannot be used for providing a solution for subscription, or for subscribing to a solution that is provided by another subaccount.

- Console client command: deploy-mta [page 1429]
  By using the `deploy-mta` command, you can deploy one or several solutions in one run.

Provided Solution

This is a solution that is deployed to the current subaccount, but provided for subscription to another SAP BTP subaccount. Before the deployment of your solution, you have to set it as a provided solution. After that you have to grant entitlements to a given SAP BTP global account that will allow its subaccounts to subscribe to the solutions.

When providing a solution for subscription, you can define which parts of it will be deployed to your subaccount, and which parts will be deployed to the subscriber's subaccount. Note that the parts deployed to your subaccount will consume resources from your quotas. All parts deployed to the subaccount of the subscriber will consume resources from its own quotas.

You can provide a solution for subscription using the:

- Cockpit: Deploy a Provided Solution [page 1095]
  You can provide a solution for subscription by selecting the *Provider deploy* checkbox in the *Deploy a Solution from an MTA Archive* dialog box that appears when you start a deployment process..
- Console client command: deploy-mta [page 1429]
  By using the `deploy-mta` command, you can provide a solution for subsription, using the `providerImport` value of the `-mode` parameter.

Available Solutions

This is a solution that is available for subscription. It has been provided by another SAP BTP subaccount and you have granted entitlements to subscribe to it. After subscribing to the solution, you can use it.

You can list the solutions that are available for subscription using the:

- Cockpit: Subscribe to a Solution Available for Subscription [page 1099]
  By using the cockpit, you can deploy all of the solution artifacts to your subaccount in one go.

- Console client command: list-mtas [page 1497]
  By using the `--available-for-subscription` parameter of the `list-mtas` command, you can list all solutions that are available for subsription to your subaccount.

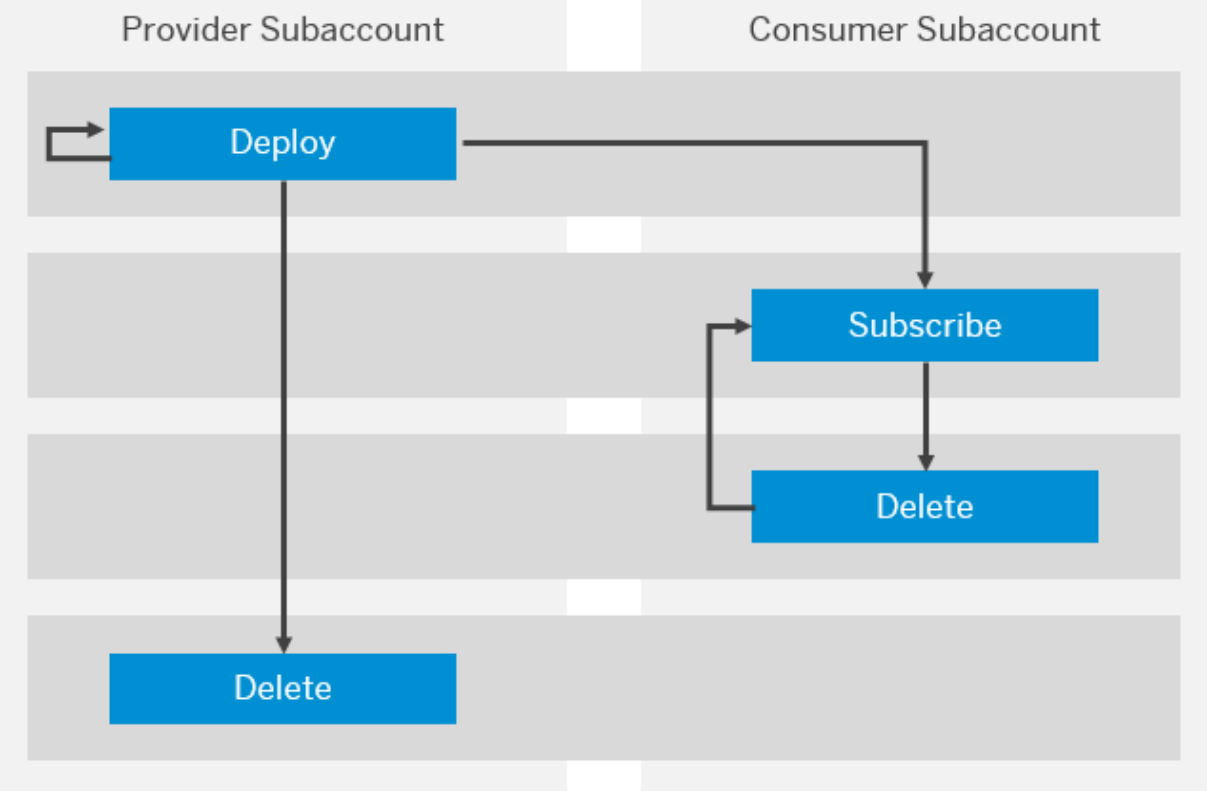
Subscribed Solution

This is a solution that has been provided by another SAP BTP subaccount. You have subscribed to it, and thus have a limited set of management operations.

When providing a solution for subscription, the provider defines which parts of it are deployed to your subaccount, and which parts are deployed to the provider subaccount. Note that the parts deployed to your subaccount consume resources from your quotas. All parts deployed to the provider subaccount consume resources from its own quotas.

You can list the solutions that are available for subscription using the:

- Cockpit: Subscribe to a Solution Available for Subscription [page 1099]
  By using the cockpit, you can deploy all of the solution artifacts to your subaccount in one go.
- Console client command: subscribe-mta [page 1568]
  By using the `subscribe-mta` command, you can subscribe to a solution that is available for subsription to your subaccount.



Operations Lifecycle of a Solution

Updating Solutions

- Using the cockpit: Updating Solutions [page 1105]

You can update solutions that have already been deployed to your subaccount by using the option in the dedicated solution overview page. Depending on the deployer version, several distinctions exist, as described in General Information About Solution Updates [page 1104].

Monitoring Solutions

After deploying a solution, you can monitor it using the:

- Cockpit: Monitor Solutions Using the Cockpit [page 1113].
- Console client command: display-mta [page 1436]

Deleting Solutions

You can delete a solution using the:

- Cockpit: Delete Solutions Using the Cockpit [page 1116]
- Console client command: delete-mta [page 1414]

Related Information

Deploying Solutions Using the Cockpit [page 1092]
Transporting Multitarget Applications with CTS+ [page 1056]
deploy-mta [page 1429]
subscribe-mta [page 1568]
Updating Solutions [page 1105]
Monitor Solutions Using the Cockpit [page 1113]
Delete Solutions Using the Cockpit [page 1116]

4.4.9.1 Deploying Solutions Using the Cockpit

By using the cockpit you can provision a solution using one of the following ways:

- Deploy a Standard Solution [page 1093]- The solution is deployed in the current subaccount and subscription to it is not possible.
- Deploy a Provided Solution [page 1095]- The solution is deployed in the current subaccount, but is provided for subscription to another subaccount.
- Subscribe to a Solution Available for Subscription [page 1099]- The current subaccount is entitled to subscribe to a solution that has been provided by another subaccount.

4.4.9.1.1 Deploy a Standard Solution

You can deploy a solution that can be consumed only within your subaccount.

Prerequisites

- The MTA archive containing your solution is created according to the information in Multitarget Applications for the Neo Environment [page 1016].
- Optionally, you have created an extension description as described in Defining MTA Extension Descriptors.
- You have a valid role for your subaccount as described in Operating Solutions [page 1089].
- You have sufficient resources available in your subaccount to deploy the content of the Multitarget Application.

> **i Note**
>
> If you are performing a redeployment of an MTA, the existing components are first deleted, which means that you do not need additional available resources.

Context

Procedure

1. Log on to the cockpit and select a subaccount.
2. Choose *Solutions* in the navigation area.
3. Choose *Deploy*.
   The *Deploy a Solution from an MTA Archive* dialog box appears.
4. Navigate to the location of the MTA archive and select it.
5. (Optional) You can provide the location of an MTA extension descriptor file.

   Alternatively, as of `_schema version 3.1`, if you do not provide it and your solution has missing data required for productive use, you can enter that data manually in the dialog subsection that appears. Keep in mind that you have to input complex parameters such as lists and maps in JSON format. For example, an account-level destination parameter `additional-properties` should be a map that has a value similar to `{"additional.property.1": "1", "additional.property.2": "2"}`.

   > **i Note**
   >
   > Make sure that you do not select the *Provider deploy* checkbox. If you select it, you will provide your solution for a subscription. For more information, see Deploy a Provided Solution [page 1095].

6. Choose *Deploy* to deploy the MTA archive and the optional MTA extension descriptor to theSAP BTP.

   The *Deploy a Solution from an MTA Archive* dialog remains on the screen during the process. When the deployment is completed, a confirmation appears that the solution has been successfully deployed. If you close the dialog during deployment, you can open it again by choosing *Check Progress* of the corresponding operation, located on the *Ongoing Operations* table in the solution overview page. You can open the page by choosing the tile of the solution that is being deployed.

   > **i Note**
   >
   > If you experience issues during the process, see Troubleshooting [page 1103].

7. (Optional) When deploying against `_schema version 3.1`, if you have manually entered parameters during deployment, at the end of the process you can download an extension descriptor containing only those parameters.

   > **i Note**
   >
   > Parameters marked as security sensitive, either by default or as set in the `mtad.yaml`, are not saved to this extension descriptor.

Results

Your newly deployed solution appears in the *Standard Solutions* category in the *Solutions* page in the cockpit.

Each solution component originates from a certain MTA module or resource, which in turn can result in several solution components. That is, one MTA module or resource corresponds to given solution components.

Related Information

Multitarget Applications for the Neo Environment [page 1016]
Defining MTA Extension Descriptors
Troubleshooting [page 1103]
Monitor Solutions Using the Cockpit [page 1113]
Delete Solutions Using the Cockpit [page 1116]

4.4.9.1.2 Deploy a Multitenant Application

Using the *Solutions* view of the cockpit, you can deploy a solution locally to your subaccount and provide it for a subscription to another subaccount or you can subscribe to a solution that has been provided for subscription by another subaccount in the cockpit.

Related Information

Deploy a Provided Solution [page 1095]
Subscribe to a Solution Available for Subscription [page 1099]

4.4.9.1.2.1 Deploy a Provided Solution

You can deploy a solution locally to your subaccount and provide it for a subscription to another subaccount.

Prerequisites

- Ensure that the MTA archive containing your solution is created as described in Multitarget Applications in the Cloud Foundry Environment.
- Optionally, you have created an extension description as described in Defining MTA Extension Descriptors.

  > **i Note**
  >
  > Several extension descriptors can be additionally used after initial deployment, that is, you can extend one extension descriptor with another unlimited times. You can use this approach if you want your subscribers to define their own data.

- You have a valid role for your subaccount as described in Operating Solutions [page 1089].
- You have sufficient resources available in your subaccount to deploy the content of the Multi-Target Application.

  > **i Note**
  >
  > - If you are performing a re-deploy, the already deployed parts of the Multi-Target Application are deleted first, so you are not required to have additional resources available in your subaccount.
  > - If parts of your solution have to be deployed to the subscribers subaccounts, note that those parts consume the resources of those subaccounts.

Context

Procedure

1. Log on to the cockpit and select a subaccount.
2. Choose *Solutions* in the navigation area.

3. Choose *Deploy*.

   The *Deploy a Solution from an MTA Archive* dialog box appears.
4. Navigate to the location of the MTA archive and select the one you want to deploy.
5. (Optional) You can provide the location of an MTA extension descriptor file.

   Alternatively, as of `_schema version 3.1`, if you do not provide it and your solution has missing data required for productive use, you can enter that data manually in the dialog subsection that appears. Keep in mind that you have to input complex parameters such as lists and maps in JSON format. For example, an account-level destination parameter `additional-properties` should be a map that has a value similar to **`{"additional.property.1": "1", "additional.property.2": "2"}`**.
6. Select the *Provider deploy* checkbox.
7. Choose *Deploy* to deploy the MTA archive and the optional MTA extension descriptor to the cloud platform.

   The *Deploy a Solution from an MTA Archive* dialog remains on the screen while the deployment is in progress. When the deployment is completed, a confirmation appears that the solution has been successfully deployed. If you close the dialog during deployment, you can open it again by choosing *Check Progress* of the corresponding operation, located on the *Ongoing Operations* table in the Solution overview page. You can open the page by choosing the tile of the solution that is being deployed.

   > **i Note**
   >
   > If you experience issues during the deployment process, see Troubleshooting [page 1103].
8. (Optional) When deploying against `_schema version3.1`, if you have manually entered parameters during deployment, at the end of the process you can use the option to download an extension descriptor containing only those parameters.

   > **i Note**
   >
   > Parameters marked as security sensitive, either by default or as set in the `mtad.yaml`, are not saved to this extension descriptor.

Results

Your newly deployed solution appears in the *Solutions Provided for Subscription* category in the *Solutions* page in the cockpit. Each solution component originates from a certain MTA module or resource that in turn can result in several solution components. That is, one MTA module or resource corresponds to given solution components.

> **i Note**
>
> If you want to create an MTA extension descriptor, for the extension ID you have to use the value of the *Extension ID* parameter, which you can find in the page of the solution you have just deployed.

Related Information

deploy-mta [page 1429]

4.4.9.1.2.1.1 Create and Edit Entitlements

After the deployment of a solution, which is going to be provided for subscription, create the entitlements that are going to be granted to the subscribers subaccounts.

Prerequisites

You have an administrator role for your subaccount.

Context

Procedure

1. Open the solution that is provided for subscription.
2. Choose *Entitlements* in the navigation area.
3. Choose *Create Entitlement*.
   The *New Entitlement* dialog box appears.
4. Enter the details for the new entitlement:
   - *Global Account ID* - the global account ID of the entitled subaccount. Consumers can obtain their global account ID by navigating to their cockpit global accounts page, and then choosing the *Show More* option on their global account tile.
     Note that by granting entitlements to a given global account, it will be accessible by all subaccounts of that global account.
   - *Effective from* - the start date from witch the provided solution is going to be available for subscription

     > **i Note**
     >
     > The default value is the current date.

   - *Granted Entitlements* - the number of subaccounts part of the global account, which are going to be able to subscribe to the provided solution

     > **i Note**
     >
     > Currently it is not possible to decrease the number of granted entitlements per particular global account.

5. Choose *Create* and then *Close*.

Results

You have created a new entitlement for a particular global account.

Edit the Number of Granted Entitlements

Prerequisites

You have an administrator role for your subaccount.

You have granted an entitlement to a particular global account.

Procedure

1. Open the solution that is provided for subscription.
2. Choose *Entitlements* in the navigation area.
3. Choose the *Edit* button under *Actions* of the entitlements you want to edit.
   The *Edit Entitlement* dialog box appears.
4. Edit the number of granted entitlements.

   > **i Note**
   >
   > Currently it is not possible to decrease the number of granted entitlements per particular global account.

5. Choose *Edit* to complete the operation.

Results

You have edited the number of granted entitlements for a particular global account.

Related Information

Defining MTA Extension Descriptors
Troubleshooting [page 1103]

Monitor Solutions Using the Cockpit [page 1113]
Delete Solutions Using the Cockpit [page 1116]

4.4.9.1.2.2 Subscribe to a Solution Available for Subscription

You can subscribe to a solution that has been provided for subscription by another subaccount in the cockpit.

Prerequisites

- You have a valid role for your subaccount as described in Operating Solutions [page 1089].
- There is a solution available for subscription in your subaccount. That is, you have been granted with an entitlement from the provider of the solution.
- You have sufficient resources available in your subaccount to deploy the content of the Multi-Target Application.

> **i Note**
>
> Typically, parts of a provided for subscription solution are deployed to the providers subaccount and parts of it within your subaccount. The parts of the solution deployed to your subaccount consume your resources, while the parts of the solution deployed to the providers subaccount consume the resources the provider subaccount.

Context

Procedure

1. Log on to the cockpit and select a subaccount.
2. Choose *Solutions* in the navigation area.
3. All solutions that you can subscribe to are listed in the *Solutions Available for Subscription* category. Select the tile of a solution that is available for subscription and then choose *Subscribe*.

   The *Subscribe to a Solution* dialog box appears.
4. (Optional) Input the location of the provider-derived MTA extension descriptor file.

   Alternatively, as of `_schema version 3.1`, if you do not provide it and your solution has missing data required for productive use, you can enter that data manually in the dialog subsection that appears. Keep in mind that you have to input complex parameters such as lists and maps in JSON format. For example, an account-level destination parameter `additional-properties` should be a map that has a value similar to `{"additional.property.1": "1", "additional.property.2": "2"}`.

> **i Note**
>
> Ensure that your extension descriptor file extends correctly the solution you are subscribing to. To do so, check the `Extension ID` of the solution in the *Additional Details* field of the solution overview page in the cockpit, and input it in the `extends` section of your extension descriptor.

5. Choose *Subscribe* to subscribe to the provided solution, and deploy the optional MTA extension descriptor to the SAP BTP.

   The *Subscribe to a Solution* dialog remains on the screen while the deployment is in progress. When the deployment is completed, a confirmation appears that the solution has been successfully deployed. If you close the dialog during deployment, you can open it again by choosing *Check Progress* of the corresponding operation, located on the *Ongoing Operations* table in the solution overview page. You can open the page by choosing the tile of the solution that is being deployed.
6. (Optional) When deploying against `_schema version3.1`, if you have manually entered parameters during deployment, at the end of the process you can use the option to download an extension descriptor containing only those parameters.

   > **i Note**
   >
   > Parameters marked as security sensitive, either by default or as set in the `mtad.yaml`, are not saved to this extension descriptor.

   > **→ Remember**
   >
   > Any resources created in the subscriber account can be updated to the provided state by resubscribing to the privided solution. You can do this either by using the *Update* option in the *Solutions* view of the SAP Cloud Platrform cockpit, or the `subscribe-mta` command of the command line interface.

Results

The solution to which you are now subscribed appears in the *Subscribed Solutions* category in the *Solutions* page in the cockpit. Each solution component originates from a certain MTA module or resource, which in turn can result in several solution components. That is, one MTA module or resource corresponds to specific solution components.

Related Information

Defining MTA Extension Descriptors
Troubleshooting [page 1103]
Monitor Solutions Using the Cockpit [page 1113]
Delete Solutions Using the Cockpit [page 1116]
subscribe-mta [page 1568]

4.4.9.2 Exporting Solutions

Prerequisites

- You have deployed or created the application components, configurations, and content that you want to export as an MTA in a SAP BTP Neo subaccount.
- Optionally, you have configured the connectivity to a transport service such as CTS+ and the Transport Management Service.

Context

You have the option to export subaccount components as Solutions. You have two possible scenarios:

- you can generate MTA development descriptor, MTA deployment descriptor template, and MTA extension descriptor template
- you can export an MTA archive and, optionally, upload it to a transport service such as CTS+ or the Transport Management Service.

> → Remember
>
> Exporting MTA archives containing Java modules is not supported. You can, however, export descriptor files that contain information about a Java software module.

The following components are supported:

- Java applications, including application destinations, data-source bindings, and role group assignments
- HTML5 applications, including permission role assignments
- Subaccount destinations
- HTML5 roles
- Cloud Portal sites
- Cloud Portal roles
- Cloud Portal destinations
- OData Provisioning service configurations
- OData Provisioning destinations
- Security groups
- Cloud Platform Integration content packages

> ℹ Note
>
> If a destination leads to a Java application and both are chosen for export, the destination `url` parameter is automatically replaced with a placeholder leading to the Java application `default-url`.

Proceed as follows:

Procedure

1. Log on to the cockpit, select a subaccount, and choose *Solutions* in the navigation area.
2. Choose *Export*. Wait until the subaccount components are discovered.
3. Choose the subaccont-level components that you want to export from the list. You can also use the search field to locate components by name. Afterward, choose *Next*.
   - The option *Automatically select dependent components* is enabled by default, so that any related components and configurations are automatically selected for you. Disable this option if you want to choose components individually.
   - The checkbox for selecting all components operates only for visible items. If you want to select all discovered items, first expand the full list by choosing *More* at the bottom of the list.
4. Deselect the subcomponents that you do not want to export. Afterward, choose *Next*.
5. Fill out the solution information. It is mandatory to enter a solution ID and a solution version. Afterward, choose *Export*.
6. Optionally, you can change the export options:
   - If you leave the default settings, upon export you are presented with download links for the MTA descriptors.
   - Your can export an MTA archive to CTS+ or the Transport Management Service. For more information the required setup, see Integration with Transport Management Tools [page 1056]. If you choose to export the MTA archive to CTS+, you can use either the TransportSystemCTS destination user or the current user to resolve the CTS+ transport request to be used.
7. Choose *Export*. As a result, you are presented with download links for the MTA development descriptor template, MTA deployment descriptor, MTA extension descriptor template, and the MTA archive. Alternatively, if in the previous step you selected the CTS+ direct export or the Transport Management Service options, the solution is also exported for usage in that system.

> → Tip
>
> You can use the MTA development descriptor template in combination with the MTA archive builder tool to create your MTA archive. The template contains `build-parameters` and `path` sections with all possible build options for the corresponding module type. Note that for your particular build environment you need to manually remove unnecessary parameters. For more information, see Cloud MTA Build Tool - Configuration .

Related Information

Operating Solutions [page 1089]
Integration with Transport Management Tools [page 1056]
Cloud MTA Build Tool
Integration with Transport Management Tools [page 1056]

4.4.9.3 Troubleshooting

While transporting SAP BTP applications using the CTS+ tool, or while deploying solutions using the cockpit, you might encounter one of the following issues. This section provides troubleshooting information about correcting them.

Troubleshooting

Message Text	Troubleshooting
`Technical error [Invalid MTA archive [<mtar archive>]. MTA deployment descriptor (META-INF/mtad.yaml) could not be parsed. Check the troubleshooting guide for guidelines on how to resolve descriptor errors. Technical details: <…>`	This error could occur if the MTA archive is not consistent. There are several different reasons for this: • The MTA deployment descriptor `META-IND/mtad.yaml` cannot be parsed, because it is syntactically incorrect according to the YAML specification. For more information, see the publicly available YAML specification. Make sure that the descriptor is compliant with the specification. Validate the descriptor syntax, for example, by using an online YAML parser. > **i Note** > Ensure that you do not submit any confidential information to the online YAML parser. • The MTA deployment descriptor might contain data that is not compatible with SAP BTP. Make sure the MTA deployment descriptor complies with the specification at Multitarget Applications for the Neo Environment [page 1016]. • The archive might not be suitable for deployment to the SAP BTP. This might happen if, for example, you attempt to deploy an archive built for XSA to the SAP BTP. The technical details might contain information similar to the following: `"Unsupported module type "<module type>" for platform type "HCP-CLASSIC""`
`Technical error [Invalid MTA archive [<MTA name>]: Missing MTA manifest entry for module [<module name>]]`	The archive is inconsistent, for example, when a module referenced in the `META-INF/mtad.yaml` is not present in the MTA archive or is not referenced correctly. Make sure that the archive is compliant with the MTA specification available at The Multitarget Application Model v.2 and the The Multitarget Application Model v.3.

Message Text	Troubleshooting
`Technical error [MTA extension descriptor(s) could not be parsed. Check the troubleshooting guide for guidelines on how to resolve descriptor errors. Technical details: <…>`	This error could occur if one or more extension descriptors are not consistent. There are several different reasons for this: • One or more extension descriptors might not be syntactically compliant with the YAML specification. Validate the descriptor syntax, for example, by using an online YAML parser. **Note** Ensure that you do not submit any confidential information to the online YAML parser. • One or more extension descriptors might contain data that is not compatible with SAP BTP. Make sure all extension descriptors comply with the specification at Multitarget Applications for the Neo Environment [page 1016].
`Technical error [MTA deployment descriptor (META-INF/mtad.yaml) from archive [<mtar archive>] and some of extension descriptors [<extension descriptor>] could not be processed . Check the troubleshooting guide for guidelines on how to resolve descriptor errors. Technical details: <…>`	This error could occur if the MTA archive, or one or more extension descriptors are not consistent. There are several different reasons for this: • The MTA deployment descriptor or an extension descriptor might contain data that is not compatible with the SAP BTP. Make sure the MTA deployment descriptor and all extension descriptors comply with the specification at Multitarget Applications for the Neo Environment [page 1016]. • The archive may not be suitable for deployment to SAP BTP. This might happen if, for example, you attempt to deploy an archive built for XSA to the SAP BTP. The technical details might contain information similar to the following: `"Unsupported module type "<module type>" for platform type "HCP-CLASSIC""`
`Process [<process-name>] has failed with [Your user is not authorized to perform the requested operation. Forbidden (403)]. Contact SAP Support.`	Ensure that you have required the necessary permissions or roles that are required to list or manage Multitarget Applications. For more information, see Operating Solutions [page 1089].
`Process [CTS_DEPLOY] has failed with [undefined (403)]. Contact SAP Support.`	

4.4.9.4 General Information About Solution Updates

To enhance your solution with new capabilities or technical improvements, you can update it using the cockpit. Depending on the deployer version (`hcp-deployer-version`) described in the MTA deployment descriptor, SAP BTP uses one of the following technical approaches, where several distinctions apply.

Redeployment

When you update your solution against deployer version `1.0` or `1.1.0`, the update is treated as a redeployment, which means:

- Any new components that have now been described in the MTA deployment descriptor are deployed as usual.
- Any already existing components are redeployed or updated, depending on their current runtime state in the SAP BTP.
- Only relations are removed to components, which are no longer present in the MTA deployment descriptor of the new solution version. The component artefacts are **not** removed.

Update with Full Semantics

When you update your solution against deployer version `1.2` or `1.2.0`, the update is treated as an update with full semantics, which means:

- Any new components that have now been described in the MTA deployment descriptor are deployed as usual
- Any already existing components are redeployed or updated, depending on their current runtime state in the SAP BTP.
- Components that are no longer present in the MTA deployment descriptor are removed.

> **i** Note
>
> The version of the MTA has to follow the “semver” Semantic Versioning specification, for example `1.1.2`.

Related Information

Updating Solutions [page 1105]
Update Options and Examples [page 1107]
http://semver.org

4.4.9.4.1 Updating Solutions

Context

You can update all solution types to a newer version.

Procedure

1. Log on to the cockpit and select the subaccount containing the solution you want to update.
2. Choose *Solutions* in the navigation area.
3. Choose the tile of the solution you want to update.
4. On the solution overview page that appears, choose *Update*.
5. **Only for standard and provided solutions**: provide the location of the MTA archive you want to use.

   > ### Note
   >
   > When you update a solution as a solution provider, ensure that the solution ID of the new deployed archive matches the ID of the existing solution.

6. (Optional) You can provide the location of an MTA extension descriptor file.

   Alternatively, as of `_schema version 3.1`, if you do not provide an MTA extension descriptor and your solution has missing data required for productive use, you can enter that data manually in the dialog subsection that appears. Keep in mind that you have to input complex parameters such as lists and maps in JSON format. For example, an account-level destination parameter `additional-properties` should be a map that has a value similar to **`{"additional.property.1": "1", "additional.property.2": "2"}`**.

7. Choose *Update* to start the process.

   > ### Note
   >
   > - Alternatively to the *Update* option, to perform the update operation you can also use the *Deploy* option.
   > - As an alternative to the cockpit procedure, you can update a solution using the following command line comand:
   >
   >   > ### Sample Code
   >   >
   >   > ```
   >   > neo deploy-mta --host <host> --account <subaccount_name>  --source
   >   > <file(1)_location>,<file(2)_location>,...,<file(n)_location> --user
   >   > <e-mail or user>
   >   > ```

Results

The existing solution is updated to a newer version.

Related Information

Defining MTA Extension Descriptors
Troubleshooting [page 1103]

Monitor Solutions Using the Cockpit [page 1113]
Delete Solutions Using the Cockpit [page 1116]
deploy-mta [page 1429]

4.4.9.4.2 Update Options and Examples

> **i Note**
>
> For the examples below we assume that you have an already deployed MTA with a deployment descriptor containing data similar to Version 1, and you want to update it to Version 2.

Example: Application Name Change While Keeping the Same Module Name

Version 1

Sample Code

```
ID: com.sap.mta.demo
_schema-version: '3.1'
modules:
 - name: administratorGroup
   parameters:
      name: &adminGroup
AdministratorGroup
   type: com.sap.hcp.group
 - name: demowebapp
   parameters:
      name: demowebapp
      jvm-arguments: -server
      title: Demo MTA Application
      runtime-version: '3'
      java-version: JRE 8
      minimum-processes: 1
      maximum-processes: 2
      roles:
       - name: administrator
         groups:
          - *adminGroup
   provides:
    - name: app_url
      properties:
         application-url: $
{default-url}
      public: true
   requires:
    - name: administratorGroup
   type: java.tomcat
 - name: demohtml5app
   parameters:
      name: demohtml5app
      version: 0.1.0-${timestamp}
   requires:
    - name: demowebapp-destination
   type: com.sap.hcp.html5
 - name: demowebapp-destination
   parameters:
      name: DemoAppBackend
      type: HTTP
      url: ~{app_url/application-
url}
      proxy-type: Internet
      authentication: AppToAppSSO
      owner: provider
   requires:
    - name: app_url
   type: com.sap.hcp.destination
parameters:
   hcp-deployer-version: '1.2.0'
   description: The application
demonstrates some of the
   main MTA features on SAP CP NEO.
   title: Demo MTA Application
   version: 0.1.1
```

Version 2

Sample Code

```
ID: com.sap.mta.demo
_schema-version: '3.1'
modules:
 - name: administratorGroup
   parameters:
      name: &adminGroup
AdministratorGroup
   type: com.sap.hcp.group
 - name: demowebapp
   parameters:
      name: demowebapp2
      jvm-arguments: -server
      title: Demo MTA Application
      runtime-version: '3'
      java-version: JRE 8
      minimum-processes: 1
      maximum-processes: 2
      roles:
       - name: administrator
         groups:
          - *adminGroup
   provides:
    - name: app_url
      properties:
         application-url: $
{default-url}
      public: true
   requires:
    - name: administratorGroup
   type: java.tomcat
 - name: demohtml5app
   parameters:
      name: demohtml5app
      version: 0.1.0-${timestamp}
   requires:
    - name: demowebapp-destination
   type: com.sap.hcp.html5
 - name: demowebapp-destination
   parameters:
      name: DemoAppBackend
      type: HTTP
      url: ~{app_url/application-
url}
      proxy-type: Internet
      authentication: AppToAppSSO
      owner: provider
   requires:
    - name: app_url
   type: com.sap.hcp.destination
parameters:
   hcp-deployer-version: '1.2.0'
   description: The application
demonstrates some of the
   main MTA features on SAP CP NEO.
   title: Demo MTA Application
   version: 0.1.2
```

In the example above, the parameter `name` of the `demowebapp` module of the MTA is changed to `demowebapp2`. This change in the MTA deployment descriptor leads to the undeployment of the Java application with name `demowebapp` and deployment of the new application `demowebapp2`. This process takes place as a key or an identifier parameter for the corresponding SAP BTP component is changed.

Example: Module Undeployment

Version 1

Sample Code

```
ID: com.sap.mta.demo
_schema-version: '3.1'
modules:
 - name: administratorGroup
   parameters:
      name: &adminGroup
AdministratorGroup
   type: com.sap.hcp.group
 - name: demowebapp
   parameters:
      name: demowebapp2
      jvm-arguments: -server
      title: Demo MTA Application
      runtime-version: '3'
      java-version: JRE 8
      minimum-processes: 1
      maximum-processes: 2
      roles:
       - name: administrator
         groups:
          - *adminGroup
   provides:
    - name: app_url
      properties:
         application-url: $
{default-url}
      public: true
   requires:
    - name: administratorGroup
   type: java.tomcat
 - name: demohtml5app
   parameters:
      name: demohtml5app
      version: 0.1.0-${timestamp}
   requires:
    - name: demowebapp-destination
   type: com.sap.hcp.html5
 - name: demowebapp-destination
   parameters:
      name: DemoAppBackend
      type: HTTP
      url: ~{app_url/application-
url}
      proxy-type: Internet
      authentication: AppToAppSSO
      owner: provider
   requires:
    - name: app_url
   type: com.sap.hcp.destination
parameters:
   hcp-deployer-version: '1.2.0'
   description: The application
demonstrates some of the
   main MTA features on SAP CP NEO.
   title: Demo MTA Application
   version: 0.1.2
```

Version 2

Sample Code

```
ID: com.sap.mta.demo
_schema-version: '3.1'
modules:
 - name: administratorGroup
   parameters:
      name: &adminGroup
AdministratorGroup
   type: com.sap.hcp.group
 - name: demowebapp2
   parameters:
      name: demowebapp2
      jvm-arguments: -server
      title: Demo MTA Application
      runtime-version: '3'
      java-version: JRE 8
      minimum-processes: 1
      maximum-processes: 2
      roles:
       - name: administrator
         groups:
          - *adminGroup
   provides:
    - name: app_url
      properties:
         application-url: $
{default-url}
      public: true
   requires:
    - name: administratorGroup
   type: java.tomcat
 - name: demowebapp-destination
   parameters:
      name: DemoAppBackend
      type: HTTP
      url: ~{app_url/application-
url}
      proxy-type: Internet
      authentication: AppToAppSSO
      owner: provider
   requires:
    - name: app_url
   type: com.sap.hcp.destination
parameters:
   hcp-deployer-version: '1.2.0'
   description: The application
demonstrates some of the
   main MTA features on SAP CP NEO.
   title: Demo MTA Application
   version: 0.1.3
```

In the example above, the module `demohtml5app` of type `com.sap.hcp.html5` is removed. This change leads to an undeplyment of the corresponding HTML5 application.

Example: New Module Deployment

Version 1

Sample Code

```
ID: com.sap.mta.demo
_schema-version: '3.1'
modules:
 - name: administratorGroup
   parameters:
      name: &adminGroup
AdministratorGroup
   type: com.sap.hcp.group
 - name: demowebapp2
   parameters:
      name: demowebapp2
      jvm-arguments: -server
      title: Demo MTA Application
      runtime-version: '3'
      java-version: JRE 8
      minimum-processes: 1
      maximum-processes: 2
      roles:
       - name: administrator
         groups:
          - *adminGroup
   provides:
    - name: app_url
      properties:
         application-url: $
{default-url}
      public: true
   requires:
    - name: administratorGroup
   type: java.tomcat
 - name: demowebapp-destination
   parameters:
      name: DemoAppBackend
      type: HTTP
      url: ~{app_url/application-
url}
      proxy-type: Internet
      authentication: AppToAppSSO
      owner: provider
   requires:
    - name: app_url
   type: com.sap.hcp.destination
parameters:
   hcp-deployer-version: '1.2.0'
   description: The application
demonstrates some of the
   main MTA features on SAP CP NEO.
   title: Demo MTA Application
   version: 0.1.3
```

Version 2

Sample Code

```
ID: com.sap.mta.demo
_schema-version: '3.1'
modules:
 - name: administratorGroup
   parameters:
      name: &adminGroup
AdministratorGroup
   type: com.sap.hcp.group
 - name: demowebapp2
   parameters:
      name: demowebapp2
      jvm-arguments: -server
      title: Demo MTA Application
      runtime-version: '3'
      java-version: JRE 8
      minimum-processes: 1
      maximum-processes: 2
      roles:
       - name: administrator
         groups:
          - *adminGroup
   provides:
    - name: app_url
      properties:
         application-url: $
{default-url}
      public: true
   requires:
    - name: administratorGroup
   type: java.tomcat
 - name: demowebapp-destination
   parameters:
      name: DemoAppBackend
      type: HTTP
      url: ~{app_url/application-
url}
      proxy-type: Internet
      authentication: AppToAppSSO
      owner: provider
   requires:
    - name: app_url
   type: com.sap.hcp.destination
 - name: demohtml5app
   parameters:
      name: demohtml5app
      version: 0.1.0-${timestamp}
   requires:
    - name: demowebapp-destination
   type: com.sap.hcp.html5

parameters:
   hcp-deployer-version: '1.2.0'
   description: The application
demonstrates some of the
   main MTA features on SAP CP NEO.
   title: Demo MTA Application
```

Version 1	Version 2
	`version: 0.1.4`

In the example above, the previously missing module `demohtml5app` is added. As a result, the corresponding HTML5 application is deployed.

Related Information

General Information About Solution Updates [page 1104]
Updating Solutions [page 1105]

4.4.9.5 Monitor Solutions Using the Cockpit

When deployed to your SAP BTP subaccount, a solution consists of various solution components. Each solution component originates from a certain MTA module that in turn can result in several solution components. That is, one MTA module corresponds to given solution components.

Prerequisites

You have a valid role for your subaccount as described in Operating Solutions [page 1089]

Context

Procedure

To see a status overview of an individual solution or solution components in your subaccount, proceed as follows:

1. Log on to the SAP BTP and select a subaccount.
2. In the cockpit, choose *Solutions* in the navigation area.

   You can monitor the overall status of the deployed and available for subscription solutions.

> **Note**
>
> The overall status of a solution is a combination of the statuses of all its internal parts and the statuses of any ongoing operations for that particular solution.

3. In the solution list, select the tile of the solution for which you want to see details.

   > **Note**
   >
   > If you have selected a solution that is available for subscription but not yet subscribed to, you can monitor only a limited set of its properties.

   The following details are available:

   - Overview - it displays the solution name and status. For more information about the solution states, see *Solutions* page help in the cockpit.
   - Description - a short descriptive text about the solution, typically stating what it does.
   - Additional Information- contains information about the provisioning type, the provider's subaccount, and the organization.
   - Ongoing Operations - the ongoing operations for the solution.
   - Solution components - a list of the components that are part of the solution, the states of these components and their types.

   For more information about the possible states of a solution component and what they mean, see your Solution page help in the cockpit.

4. If you have provided a solution that is available for subscription to another subaccount, you can monitor the licenses and subscribers of a provided solution as follows:

   a. In the solution list under the *Solutions Provided for Subscription* category, select the tile of the solution for which you want to see details.
   b. Choose *Entitlement* in the navigation area of the cockpit.

      You can monitor the granted entitlements for that solution as well as the parts that were deployed to the subscribers subaccounts.

      > **Note**
      >
      > Monitoring granted licenses is only available for you if you have the subaccount administrator role.

Results

Depending on the solution type, the following components can be monitored:

Solution Components

	Results in One or More of the Following Solution Components		
MTA Modules	**Standard Solution**	**Provided Solution**	**Subscribed Solution**
`com.sap.java`	• Java Application • Data Source Binding • SAP SuccessFactorss Connection • SAP SuccessFactors Homepage Tile • SAP SuccessFactors Application Access • SAP SuccessFactors Role Provider • Subaccount Destination	• Java Application • Database Binding • Subaccount Destination	• Java Application Subscription • SAP SuccessFactors Application Access • SAP SuccessFactors Homepage Tile • SAP SuccessFactors Role Provider • SAP SuccessFactors Connection • Subaccount Subscription Destination
`com.sap.hcp.html5`	• HTML5 Application • SAP SuccessFactors Application Access • SAP SuccessFactors Homepage Tile	HTML5 Application	• HTML5 Application Subscription • SAP SuccessFactors Application Access • SuccessFactors Homepage Tile
`com.sap.fiori`	Fiori Content	Fiori Content	Fiori Content
`com.sap.fiori.app`	HTML5 Application	HTML5 Application	HTML5 Application Subscription
`com.sap.fiori.role`	Role	Role	Role
`com.sap.odp.config`	OData Service	OData Service	OData Service
`com.sap.hcp.sfsf-roles`	SAP SuccessFactors Role		SAP SuccessFactors Role
`com.sap.hcp.destination`	Subaccount Destination	Subaccount Destination	Subaccount Destination

Related Information

Deploying Solutions Using the Cockpit [page 1092]

4.4.9.6 Delete Solutions Using the Cockpit

Delete a solution from your subaccount following the steps for the corresponding solution types.

Prerequisites

You have a valid role for your subaccount as described in Operating Solutions [page 1089]

Context

> **i Note**
>
> - Some parts of the solution might be shared and used by other entities within the SAP BTP. Such parts of the solution have to be removed manually.
> - SAP SuccessFactors roles are not deleted.
> - Custom application destinations and subaccount destinations are also deleted.
> - Deleted solutions and their components cannot be restored.
> - Currently, deleting a solution that has been provided for subscription is not automated. Subaccounts consuming your provided solutions have to delete their subscribed solutions before you delete that solution from your subaccount.
> - If the solution has been provided to you for subscription from a provider subaccount, your entitlement is not deleted. You will be able to subscribe again to the provided solution.

To delete a solution from your subaccount, proceed as described below.

Procedure

1. In the cockpit, choose *Solutions* in the navigation area.
2. In the solution list, select the tile of the solution you want to undeploy, and afterward choose *Delete*. As a result, the *Delete a Solution* dialog appears.
3. Optionally, you can perform the following:
    1. Deselect the *Delete data source bindings* checkbox.

        > **i Note**
        >
        > If the *Delete data source* checkbox is selected, any deployed database binding will be deleted. Note that your database credentials will not be removed from your database and can be used again.

    2. Select the *Clean-up on error* checkbox.

> **i Note**
>
> If set, any errors during deletion that are external to SAP BTP (for example, a SuccessFactors system) are ignored.
>
> Typical use case is to be able to delete a solution that is linked to a now nonexistent external system. Then, if the *Clean-up on error* checkbox is not selected, the deletion process will fail with an error. When the *Clean-up on error* is selected the deletion process will ignore the error and continue.

> **i Note**
>
> If the *Clean-up on error* checkbox is selected and an error that originates from an external to SAP BTP instance occurs, it will be ignored. As a result all the data stored in the SAP BTP for that solution will be deleted. However, external systems might still contain some data that is not deleted.

4. Select the *Confirm solution deletion* checkbox, and choose *Delete*.

   The solution deletion dialog remains on the screen during the process. A confirmation appears when the deletion is completed.

   If you close the dialog while the process is running, you can open it again by choosing *Check Progress* of the corresponding operation, located in the *Ongoing Operations* table in the solution overview page.

Results

The solution is deleted from the SAP BTP.

Related Information

Deploying Solutions Using the Cockpit [page 1092]

4.5 Developing HTML5 Applications in the Neo Environment

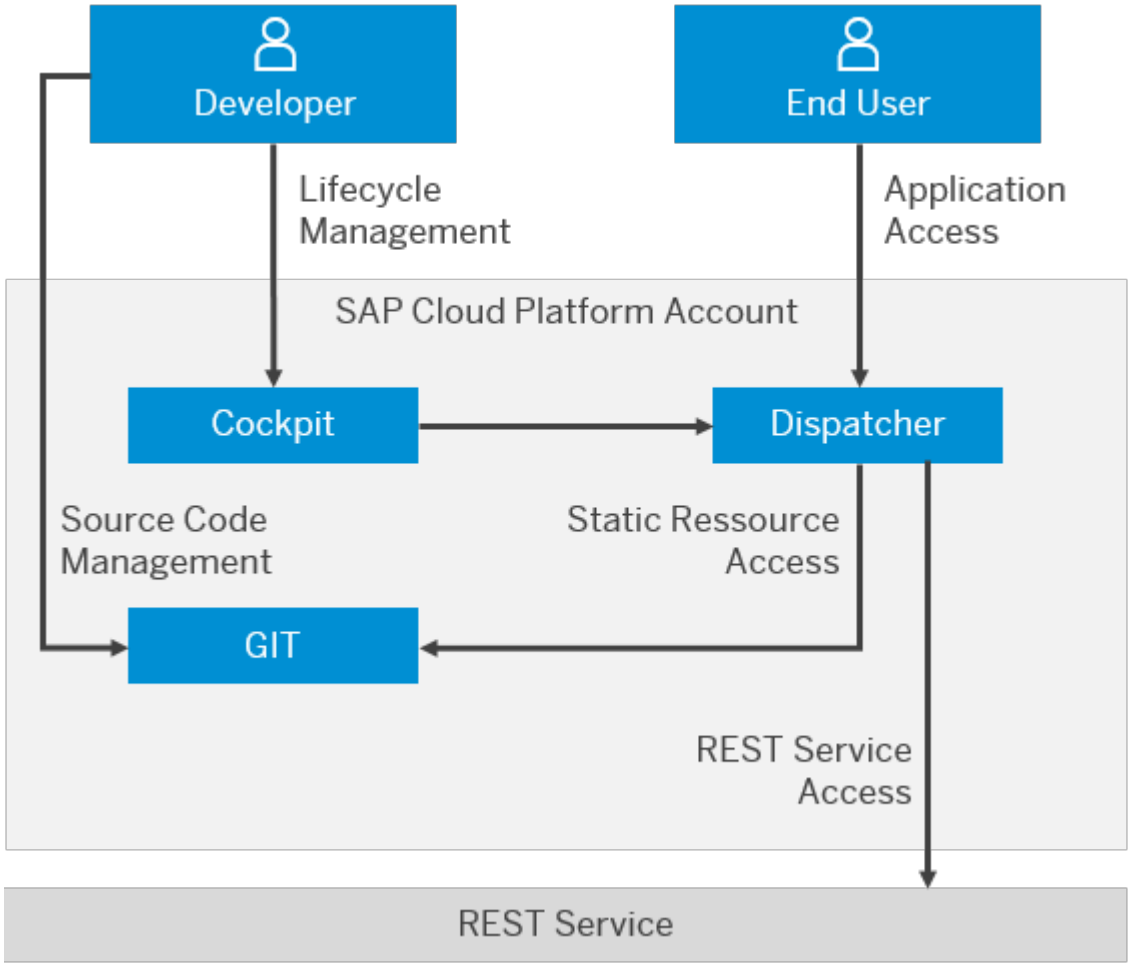
SAP BTP enables you to easily develop and run HTML5 applications in a cloud environment.

HTML5 applications on SAP BTP consist of static resources and can connect to any existing on-premise or on-demand REST services. Compared to a Java application, there is no need to start a dedicated process for an HTML5 application. Instead the static resources and REST calls are served using a shared dispatcher service provided by the SAP BTP.

The static content of the HTML5 applications is stored and versioned in Git repositories. Each HTML5 application has its own Git repository assigned. For offline editing, developers can directly interact with the Git

service using a Git client of their choice. They may use any Git client like EGit or a native Git implementation to perform Git operations. A Git repository is created automatically when a new HTML5 application is created.

Lifecycle operations, for example, creating new HTML5 applications, creating new versions, activating, starting and stopping or testing applications, can be performed using the SAP BTP cockpit. As the static resources are stored in a versioned Git repository, not only the latest version of an application can be tested, but the complete version history of the application is always available for testing. The version that is delivered to the end users of that application is called the "active version". Each application can have only one active version.



HTML5 Applications Overview

Related Information

HTML5: Getting Started [page 1119]
HTML5: Application Operations [page 1631]
Securing HTML5 Applications [page 1744]

4.5.1 HTML5: Getting Started

Set up your HTML5 development environment and run your first application in the cloud.

1. Getting a Global Account
   You first need to sign up for an SAP BTP subaccount.
2. Add Users
   Add users who develop and maintain HTML5 applications as subaccount members of your subaccount.
3. Set Up
   To develop HTML5 applications, we recommend that you use the browser-based tool SAP Web IDE that does not require any setup.
4. Create
   Create a simple HTML5 application and run it in the cloud: Creating a Hello World Application Using SAP Web IDE [page 1119]

For more information about building applications in SAP Web IDE, see the SAP Web IDE documentation. There, you will also find information on building your project first and then pushing your app to the cockpit.

Related Information

Developing HTML5 Applications in the Neo Environment [page 1117]
HTML5: Application Operations [page 1631]
Securing HTML5 Applications [page 1744]

4.5.1.1 Creating a Hello World Application Using SAP Web IDE

This tutorial illustrates how to build a simple HTML5 application using SAP Web IDE.

Prerequisites

- Your company has signed up for an SAP BTP subaccount.
- You are a member of the SAP BTP subaccount.

Context

During the tutorial, you perform the following tasks:

1. Create an HTML5 Application [page 1120]
   You create new applications in the SAP BTP cockpit.
2. Create a Project [page 1121]
   A project is needed to create files and to make them available in the cockpit.
3. Edit the HTML5 Application [page 1122]
   SAP Web IDE already created an HTML page for your project. You now adapt this page.
4. Deploy Your App to SAP BTP [page 1123]
   With this step you create a new active version of your app that is started on SAP BTP.

4.5.1.1.1 Create an HTML5 Application

You create new applications in the SAP BTP cockpit.

Context

For each new application a new Git repository is created automatically. To view detailed information on the Git repository, including the repository URL and the latest commits, choose *Applications* > *HTML5 Applications* in the navigation area and then *Versioning*.

> **i Note**
>
> To create the HTML5 application in more than one region, create the application in each region separately and copy the content to the new Git repository.

Procedure

1. Log on with a user (who is a subaccount member) to the SAP BTP cockpit.
2. Choose *Applications* > *HTML5 Applications* in the navigation area.

   If you have already created applications using this subaccount, the list of HTML5 applications is displayed.
3. To create a new HTML5 application, choose *New Application* and enter an application name.

   > **i Note**
   >
   > Adhere to the naming convention for application names:
   >
   > - The name must contain no more than 30 characters.
   > - The name must contain only lowercase alphanumeric characters.
   > - The name must start with a letter.
4. Choose *Save*.
5. Clone the repository to your development environment.

a. To start SAP Web IDE and automatically clone the repository of your app, choose *Edit Online* ( ) at the end of the table row of your application.
b. On the *Clone Repository* screen, if prompted enter your user and password (SCN user and SCN password), and choose *Clone*.

Results

You created an application and a corresponding Git repository.

Task overview: Creating a Hello World Application Using SAP Web IDE [page 1119]

Next task: Create a Project [page 1121]

Related Information

Navigate in the Cockpit

4.5.1.1.2 Create a Project

A project is needed to create files and to make them available in the cockpit.

Context

Procedure

1. In SAP Web IDE, choose *Development* (*</>*), and then select the project of the application you created in the cockpit.
2. To create a project and to clone your app to the development environment, right-click the project, and choose *New* > *Project from Template* .
3. Choose the *SAPUI5 Application* button, and choose *Next*.
4. In the *Project Name* field, enter a name for your project, and choose *Next*.

> **i Note**
>
> The project name has to be unique within the workspace.

5. Fill in the following fields, and then choose *Next*:

Field	Entry
View Type	Select *JavaScript*.
View Name	Enter **`HelloWorld`** (without spaces).

6. Choose *Finish*.

Task overview: Creating a Hello World Application Using SAP Web IDE [page 1119]

Previous task: Create an HTML5 Application [page 1120]

Next task: Edit the HTML5 Application [page 1122]

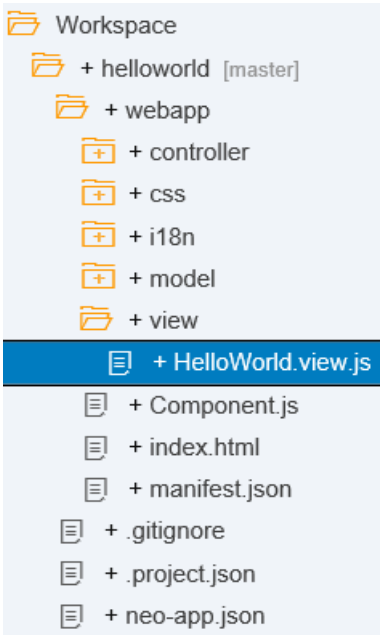
4.5.1.1.3 Edit the HTML5 Application

SAP Web IDE already created an HTML page for your project. You now adapt this page.

Context

Procedure

1. In SAP Web IDE, expand the project node in the navigation tree and open the `HelloWorld.view.js` using a double-click.



2. In the `HelloWorld.view.js` view, replace `Title` in the `title: "{i18n>title}"` line with the title of your application **`Hello World`**.
3. Save your changes using *Save* (

).
4. To test your Hello World application, select the `index.html` file and choose *Run* (

).

Task overview: Creating a Hello World Application Using SAP Web IDE [page 1119]

Previous task: Create a Project [page 1121]

Next task: Deploy Your App to SAP BTP [page 1123]

4.5.1.1.4 Deploy Your App to SAP BTP

With this step you create a new active version of your app that is started on SAP BTP.

Context

Procedure

1. In SAP Web IDE, select the project node in the navigation tree.
2. To deploy the project, right-click it and choose *Deploy* > *Deploy to SAP BTP* .
3. On the *Login to SAP BTP* screen, enter your password and choose *Login*.
4. On the *Deploy Application to SAP BTP* screen, increment the version number and choose *Deploy*.

> **i** Note
>
> If you leave the *Activate* option checked, the new version is activated directly.

5. Confirm the success message with *OK*.

Task overview: Creating a Hello World Application Using SAP Web IDE [page 1119]

Previous task: Edit the HTML5 Application [page 1122]

4.5.2 Developer's Guide

The developer's guide introduces the development environment for HTML5 applications, a procedure on how to create applications, and supplies details on the descriptor file that specifies how dedicated application URLs are handled by the platform.

Related Information

Development Environment [page 1124]
Create an HTML5 Application [page 1126]
Application Descriptor File [page 1128]

4.5.2.1 Development Environment

The development workflow is initiated from the SAP BTP cockpit.

The cockpit provides access to all lifecycle operations for HTML5 applications, for example, creating new applications, creating new versions, activating a version, and starting or stopping an application.

4.5.2.1.1 Working with Git

The SAP Git service stores the sources of an HTML5 application in a Git repository.

For each HTML5 application there is one Git repository. You can use any Git client to connect to the Git service. On your development machine you may, for example, use Native Git or Eclipse/EGit. The SAP Web IDE has a built-in Git client.

Git URL

With this URL, you can access the Git repository using any Git client.

The URL of the Git repository is displayed under *Source Location* on the detail page of the repository. You can also view this URL together with other detailed information on the Git repository, including the repository URL and the latest commits, by choosing *HTML5 Applications* in the navigation area and then *Versioning*.

Authentication

Access to the Git service is only granted to authenticated users. Any user who is a member of the subaccount that contains the HTML5 application and who has the *Administrator*, *Developer*, or *Support User* role has

access to the Git repository. When sending requests, users have to authenticate themselves using their user name and password of the identity provider.

Permissions

The permitted actions depend on the subaccount member role of the user:

Any authenticated user with the *Administrator*, *Developer*, or *Support User* role can read the Git repository. They have permission to:

- Clone the repository.
- Fetch commits and tags.

Write access is granted to users with the *Administrator* or *Developer* role. They have permission to:

- Push commits if the branch pointer on the server is moved fast-forward.
  Otherwise, the commits need to be rebased locally so that they are successors of the tip of the branch the push is targeting.
- Push tags, if no tag with the same name exists.
  Pushing a tag defines a new version of the HTML5 application. The version name is the same as the tag name.
- Create a new branch.
  Currently, only commits in the master branch are visible in the SAP BTP cockpit.
- Push commits initiated by a different author (forge author identity).
  Developers are not allowed to push commits committed by a different user (forge committer identity). The committer e-mail address in the commits they push must match the e-mail address they registered in the identity provider.
- Developers cannot delete or move tags or delete branches.

Only users with the *Administrator* role have permission to:

- Push commits committed by a different user (forge committer identity).
- Forcefully push commits, for example, to rewrite the Git history of an HTML5 application.
- Forcefully push tags, for example, to move the version of an HTML5 application to a different commit.
- Delete tags or delete branches.

Related Information

Managing Member Authorizations in the Neo Environment [page 1301]

4.5.2.2 Create an HTML5 Application

You create new applications in the SAP BTP cockpit.

Context

For each new application a new Git repository is created automatically. To view detailed information on the Git repository, including the repository URL and the latest commits, choose *Applications* > *HTML5 Applications* in the navigation area and then *Versioning*.

> **i Note**
>
> To create the HTML5 application in more than one region, create the application in each region separately and copy the content to the new Git repository.

Procedure

1. Log on with a user (who is a subaccount member) to the SAP BTP cockpit.
2. Choose *Applications* > *HTML5 Applications* in the navigation area.

   If you have already created applications using this subaccount, the list of HTML5 applications is displayed.
3. To create a new HTML5 application, choose *New Application* and enter an application name.

   > **i Note**
   >
   > Adhere to the naming convention for application names:
   >
   > - The name must contain no more than 30 characters.
   > - The name must contain only lowercase alphanumeric characters.
   > - The name must start with a letter.
4. Choose *Save*.
5. Clone the repository to your development environment.
   a. To start SAP Web IDE and automatically clone the repository of your app, choose *Edit Online* ( ) at the end of the table row of your application.
   b. On the *Clone Repository* screen, if prompted enter your user and password (SCN user and SCN password), and choose *Clone*.

Results

You created an application and a corresponding Git repository.

Related Information

Navigate in the Cockpit

4.5.2.3 Create a Version

You create a version of your application from one of the commits.

Context

The version is a tag that is attached to the commit.

Procedure

1. Log on with a user (who is a subaccount member) to the SAP BTP cockpit.
2. Choose the *Applications* > *HTML5 Applications* in the navigation area.
3. Select your Hello World application.
4. Switch to the *Versioning* section.
5. In the *Available Commits* table, select the commit you want to use and choose the *Create Version* icon.
6. Enter a version name and choose *Add*.

Results

You can now activate this version to make the application available to the end users.

Related Information

Navigate in the Cockpit

4.5.2.4 Activate a Version

As end users can only access the active version of an application, you must create and activate a version of your application.

Context

The developer can activate a single version of an application to make it available to end users.

Procedure

1. Log on with a user (who is a subaccount member) to the SAP BTP cockpit.
2. Choose the *Applications* > *HTML5 Applications* in the navigation area.
3. Select your Hello World application.
4. Switch to the *Versioning* section.
5. Choose the *Versions* button.
6. In the *Versions* table, select your version and choose the *Activate this application version* icon.
7. Confirm that you want to activate the application.

Results

You can now distribute the URL of your application to the end users.

Related Information

Navigate in the Cockpit

4.5.2.5 Application Descriptor File

Using the application descriptor file you can configure the behavior of your HTML5 application.

This descriptor file is named `neo-app.json`. The file must be created in the root folder of the HTML5 application repository and must have a valid JSON format.

With the descriptor file you can set the options listed under *Related Links*.

The application descriptor file must follow the following format. All settings are optional.

```
{
    "authenticationMethod": "saml"|"none",
    "welcomeFile": "<path to welcome file>",
    "logoutPage": "<path to logout page>",
    "sendWelcomeFileRedirect": true|false,
    "routes": [
        {
            "path": "<application path to be mapped>",
            "target": {
                "type": "destination | service | application",
                "name": "<name of the destination> | <name of the service> |
<name of the application or subscription>",
                "entryPath": "<path prepended to the request path>",
                "version": "<version to be referenced. Default is active
version.>"
            },
            "description": "<description>"
        }
    ],
    "securityConstraints": [
        {
            "permission": "<permission name>",
            "description": "<permission description>",
            "protectedPaths": [
                "<path to be secured>",
                 ...
            ],
            "excludedPaths": [
                "<path to be excluded>",
                 ...
            ]
        }
    ],
    "cacheControl": [
        {
            "path": "<optional path of resources to be cached>",
            "directive": "none | public | private",
            "maxAge": <lifetime in seconds>
        }
    ],
    "headerWhiteList": [
        "<header1>",
        "<header2>",
         ...
    ]
}
```

All paths in the `neo-app.json` must be specified as plain paths, that is, paths with blanks or other special characters must include these characters literally. These special characters must be URI-encoded in HTTP requests.

Related Information

Information about: authenticationMethod [page 1130]
Information about: welcomeFile and sendWelcomeFileRedirect [page 1142]
Information about: routes with target type destination [page 1134]
Information about: routes to include SAPUI5 resources [page 1133]
Information about: routes to access the user API [page 1139]

Information about: securityConstraints [page 1131]
Information about: cacheControl [page 1143]
Information about: headerWhiteList [page 1144]
Accessing Application Resources [page 1138]

4.5.2.5.1 Authentication

Authentication is the process of establishing and verifying the identity of a user as a prerequisite for accessing an application.

By default an HTML5 application is protected with SAML2 authentication, which authenticates the user against the configured IdP. For more information, see Application Identity Provider [page 1720]. For public applications the authentication can be switched off using the following syntax:

```
"authenticationMethod": "saml" | "none"
```

Example

An example configuration that switches off authentication looks like this:

```
"authenticationMethod": "none"
```

Note

Even if authentication is disabled, authentication is still required for accessing inactive application versions.

To protect only parts of your application, set the `authenticationMethod` to `"none"` and define a security constraint for the paths you want to protect. If you want to enforce only authentication, but no additional authorization, define a security constraint without a permission (see Authorization [page 1131]).

Handling Session Timeout

After 20 minutes of inactivity user sessions are invalidated. If the user tries to access an invalidated session, SAP BTP returns a logon page, where the user must log on again. If you are using SAML as a logon method, you cannot rely on the response code to find out whether the session has expired because it is either 200 or 302. To check whether the response requires a new logon, get the `com.sap.cloud.security.login` HTTP header and reload the page. For example:

```
jQuery(document).ajaxComplete(function(e, jqXHR) {
    if(jqXHR.getResponseHeader("com.sap.cloud.security.login")) {
        alert("Session is expired, page shall be reloaded.");
        window.location.reload();
    }
})
```

4.5.2.5.2 Authorization

To enforce authorization for an HTML5 application, permissions can be added to application paths.

In the cockpit, you can create custom roles and assign them to the defined permissions. If a user accesses an application path that starts with a path defined for a permission, the system checks if the current user is a member of the assigned role. If no role is assigned to a defined permission only subaccount members with developer permission or administrator permission have access to the protected resource.

Permissions are only effective for the active application version. To protect non-active application versions, the default permission `NonActiveApplicationPermission` is defined by the system for every HTML5 application. This default permission must not be defined in the `neo-app.json` file but is available automatically for each HTML5 application.

If only authentication is required for a path, but no authorization, a security constraint can be added without a permission.

A security constraint applies to the directory and its sub-directories defined in the `protectedPaths` field, except for paths that are explicitly excluded in the `excludedPaths` field. The `excludedPath` field supports pattern matching. If a path specified ends with a slash character (`/`) all resources in the given directory and its sub-directories are excluded. You can also specify the path to be excluded using wildcards, for example, the path `**.html` excludes all resources ending with `.html` from the security constraint.

To define a security constraint, use the following format in the `neo-app.json` file:

```
...
    "securityConstraints": [
        {
            "permission": "<permission name>",
            "description": "<permission description>",
            "protectedPaths": [
                "<path to be secured>"
            ],
            "excludedPaths": [
                "<path to be excluded>",
                 ...
            ]
        }
    ]
...
```

Example

An example configuration that restricts a complete application to the `accessUserData` permission, with the exception of all paths starting with `"/logout"`, looks like this:

```
...
    "securityConstraints": [
        {
            "permission": "accessUserData",
            "description": "Access User Data",
            "protectedPaths": [
                "/"
            ],
            "excludedPaths": [
                "/logout/**"
            ]
        }
    ]
...
```

Related Information

Managing Roles and Permissions [page 1637]

4.5.2.5.3 Protecting the Application Descriptor

By default end users can access the application descriptor file of an HTML5 application.

To do so, they enter the URL of the application followed by the filename of the application descriptor in the browser.

> → Tip
>
> For security reasons we recommend that you use a permission to protect the application descriptor from being accessed by end users.

A permission for the application descriptor can be defined by adding the following security constraint into the application descriptor

```
...
    "securityConstraints": [
         {
              "permission": "AccessApplicationDescriptor",
              "description": "Access application descriptor",
              "protectedPaths": [
                 "/neo-app.json"
              ]
         }
    ]
...
```

After activating the application, a role can be assigned to the new permission in the cockpit to give users with that role access to the application descriptor via the browser. For more information about how to define permissions for an HTML5 application, see Authorization [page 1131].

4.5.2.5.4 Accessing SAPUI5 Resources

To access SAPUI5 resources in your HTML5 application, configure the SAPUI5 service routing in the application descriptor file.

SAPUI5 Service Routing

To configure the SAPUI5 service routing for your application, map a URL path that your application uses to access SAPUI5 resources to the SAPUI5 service:

```
...
"routes": [
        {
           "path": "<application path to be mapped>",
           "target": {
               "type": "service",
               "name": "sapui5",
               "version": "<version>",
               "entryPath": "/resources"
                     },
           "description": "<description>"
           }
        ]
...
```

Example

This configuration example maps all paths starting with `/resources` to the `/resources` path of the SAPUI5 library.

```
...
"routes": [
        {
           "path": "/resources",
           "target": {
               "type": "service",
               "name": "sapui5",
               "entryPath": "/resources"
                     },
           "description": "SAPUI5"
           }
        ]
...
```

For more information about using SAPUI5 for your application, see SAPUI5: UI Development Toolkit for HTML5.

Determining the Current Active SAPUI5 Version

In the top right corner on **`https://sapui5-sapui5.dispatcher.hana.ondemand.com/`**, choose *VERSION INFO* to display the latest versions of the individual libs.

Referencing a Specific SAPUI5 Version in Your HTML5 App

> **Example**
>
> This configuration example shows how to reference the SAPUI5 version 1.26.6 using the `neo-app.json` file.

```
...
"routes": [
    {
        "path": "/resources",
        "target": {
          "type": "service",
          "name": "sapui5",
          "version": "1.26.6",
          "entryPath": "/resources"
        },
        "description": "SAPUI5"
    }
}
...
```

To display the available SAPUI5 versions, open **`https://sapui5-sapui5.dispatcher.hana.ondemand.com/neo-app.json`** file.

Related Information

VERSION INFO in SAPUI5 Release Notes

To view the SAPUI5 versions in the HTML5 App, open the `neo-app.json` file

Release Notes for SAP BTP

4.5.2.5.5 Accessing REST Services

To connect your application to a REST service, configure routing to an HTTP destination in the application descriptor file.

A route defines which requests to the application are forwarded to the destination. Routes are matched with the path from a request. All requests with paths that start with the path from the route are forwarded to the destination.

If you define multiple routes in the application descriptor file, the route for the first matching path is selected.

The HTTP destination must be created in the subaccount where the application is running. For more information on HTTP destinations, see Create HTTP Destinations [page 80] and Assign Destinations for HTML5 Applications [page 1636].

```
...
"routes": [
          {
             "path": "<application path to be forwarded>",
```

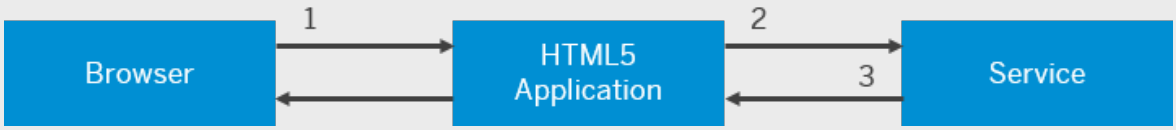
```
              "target": {
                   "type": "destination",
                   "name": "<name of the destination>"
              },
             "description": "<description>"
           }
         ]
...
```

Example

With this configuration, all requests with paths starting with `/gateway` are forwarded to the `gateway` destination.

```
...
"routes": [
           {
              "path": "/gateway",
              "target": {
                   "type": "destination",
                   "name": "gateway"
              },
             "description": "Gateway System"
           }
         ]
...
```

The browser sends a request to your HTML5 application to the path `/gateway/resource` (1). This request is forwarded by the HTML5 application to the service behind the destination `gateway` (2). The path is shortened to `/resource`. The response returned by the service is then routed back through the HTML5 application so that the browser receives the response (3).



⚠ Caution

Routes to destinations using the web socket protocol is not supported.

Destination Properties

In addition to the application-specific setup in the application descriptor, you can configure the behavior of routes at the destination level. For information on how to set destination properties, see You can enter additional properties (step 9) [page 80].

You can set the following properties:

Property	Type/Unit	Default	Maximum	Recommended Value	Description
`HTML5.ConnectionTimeoutInSeconds`	Integer/seconds	10	120	Default	The period of time in which the HTML5 application expects a confirmation when it initiates a new TCP connection.
`HTML5.SocketReadTimeoutInSeconds`	Integer/seconds	30	300	Default	The period of time the HTML5 application waits for the REST service to start streaming the response.
`HTML5.HandleRedirects`	Boolean	`true`	--	`false`	When set to `true` the HTML5 application internally calls the target URL of a redirecting HTTP response (HTTP 30X status code). Then, the browser will only see this second response.
`sap-client`	String	--	--	--	If `true`, the HTML5 Applications propagates the sap-client and its value as a header in the back-end request. This is expected by ABAP back-end systems.

Timeout Handling

A request to a REST service can time out when the network or backend is overloaded or unreachable. Different timeouts apply for initially establishing the TCP connection (`HTML5.ConnectionTimeoutInSeconds`) and reading a response to an HTTP request from the socket (`HTML5.SocketReadTimeoutInSeconds`). When a timeout occurs, the HTML5 application returns a gateway timeout response (`HTTP status code 504`) to the client.

While some long-running requests may require to increase the socket timeout, we do not recommend that you change the default values. Too high timeouts may impact the overall performance of the application by blocking other requests in the browser or blocking back-end resources.

Redirect Handling

By default all HTML5 applications follow HTTP redirects of REST services internally. This means whenever your REST service responds with a `301`, `302`, `303`, or `307` HTTP status code, a new request is issued to the redirect target. Only the response to this second request reaches the browser of the end user. To change this behavior,

set the `HTML5.HandleRedirects` destination property to `false`. As a consequence, the 30X responses given above are directly sent back without following the redirect.

We recommend that you set this property to `false`. This helps improve the performance of your HTML5 application because the browser stores redirects and thus avoids round trips. If you use relative links, the automatic handling of redirects might break your HTML5 application on the browser side. However, certain service types may not run with a value of `false`.

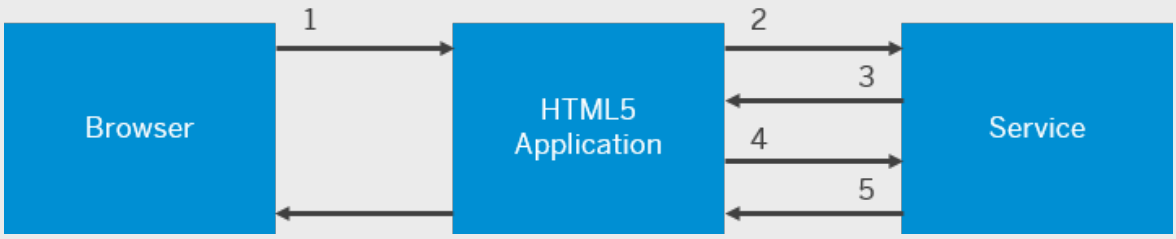
Example

Prerequisites:

- Your application descriptor contains a route that forwards requests starting with the path `/gateway`, to the destination named `gateway` as in the example above.
- The service redirects requests from `/resource` to the path `./servicePath/resource`.

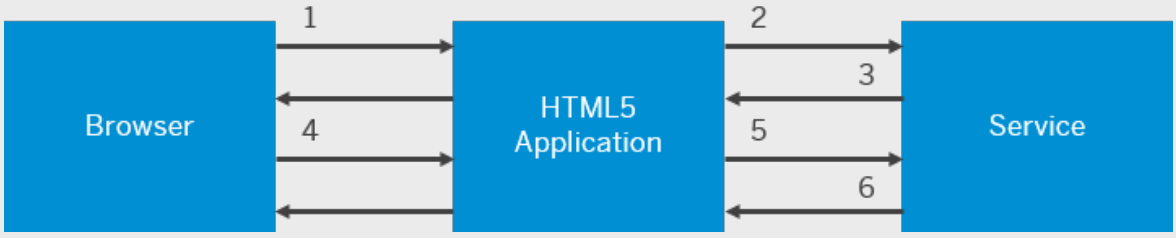
`HTML5.HandleRedirects` is set to `true`:

When the browser requests the path `/gateway/resource` (1), the HTML5 application forwards it to the path `/resource` of the service (2). As the service responds with a redirect (3), the HTML5 application sends another request to the new path `/servicePath/resource` (4). This second response contains the required resource and is forwarded back to the browser (5).



`HTML5.HandleRedirects` is set to `false`:

For the same request to the path `/gateway/resources` (1), the HTML5 application again forwards the request to the path `/resources` of the service (2). Now the redirect is directly forwarded back to the browser (3). In this case it is the browser that sends another request to the path `/gateway/servicePath/resource` (4), which the HTML5 application forwards to the service path `/servicePath/resource` (5). The requested resource is then forwarded back to the browser (6).



Deprecated Properties

The following destination properties have been deprecated and replaced by new properties. If the new and the old properties are both set, the new property overrules the old one.

Old Property	Type/Unit	Values	Replaced by New Property
`ConnectionTimeout`	Integer/milliseconds	Default: `10000` (10 seconds) Max. value: `120000` (120 seconds)	`HTML5.ConnectionTimeoutInSeconds`
`ClientReadTimeout`	Integer/milliseconds	Default: `30000` (30 seconds) Max. value: `300000` (300 seconds)	`HTML5.SocketReadTimeoutInSeconds`

Security Considerations

When accessing a REST service from an HTML5 application, a new connection is initiated by the HTML5 application to the URL that is defined in the HTTP destination.

To prevent that security-relevant headers or cookies are returned from the REST service to the client, only whitelisted headers are returned. While some headers are whitelisted per default, additional headers can be whitelisted in the application descriptor file. For more information about how to whitelist additional headers, see Approving HTTP Headers [page 1144].

Cookies that are retrieved from a REST service response are stored by the HTML5 application in an HTTP session that is bound to the client request. The cookies are not returned to the client. If a subsequent request is initiated to the same REST service, the cookies are added to the request by the application. Only those cookies are added that are valid for the request in the sense of correct domain and expiration date. When the client session is terminated, all associated cookies are removed from the HTML5.

Related Information

Assign Destinations for HTML5 Applications [page 1636]

4.5.2.5.6 Accessing Application Resources

To access resources from another HTML5 application or a subscription to an HTML5 application, you can map an application path to the corresponding application or subscription.

If the given path matches a request path, the resource is loaded from the mapped application or subscription. This feature may be used to separate re-usable resources in a dedicated application.

> **Note**
>
> If multiple routes are defined in the application descriptor, the route for the first matching path in the application descriptor is selected.

```
...
"routes": [
        {
           "path": "<application path to be mapped>",
           "target": {
               "type": "application",
               "name": "<name of the application or subscription>"
               "version": "<version to be referenced. Default is active
version>",
               },
           "description": "<description>"
             }
          ]
...
```

> **Example**
>
> This configuration example maps all paths starting with `/icons` to the active version of the application named `iconlibrary`.

```
...
"routes": [
             {
           "path": "/icons",
           "target": {
               "type": "application",
               "name": "iconlibrary"
               },
           "description": "Icon Library"
             }
          ]
...
```

Related Information

Application Descriptor File [page 1128]

4.5.2.5.7 Accessing the User API

The User API service provides an API to query the details of the user that is currently logged on to the HTML5 application.

If you use a corporate identity provider (IdP), some features of the API do not work as described here. The corporate IdP requires you to configure a mapping from your IdP's assertion attributes to the principal attributes usable in SAP BTP. See Configure User Attribute Mappings [page 1727].

To use the User API in your HTML5 application, add a route to your `neo-app.json` application descriptor file as follows:

```
...
"routes": [
   {
     "path": "<application path to be forwarded>",
     "target": {
       "type": "service",
       "name": "userapi"
     }
  }
]
...
```

The route defines which requests to the application are forwarded to the API. The route is matched with the path from a request. All `GET` requests with paths that start with the `path` from the route are forwarded to the API.

Example

With the following configuration, all `GET` requests with paths starting with `/services/userapi` are forwarded to the user API.

```
...
"routes": [
   {
     "path": "/services/userapi",
     "target": {
       "type": "service",
       "name": "userapi"
     }
  }
]
...
```

The User API supports the following endpoints:

- `/currentUser`
- `/attributes`

The User API requires authentication. The user is logged on automatically even if the `authentication` property is set to `none` in the `neo-app.json` file.

Current User Details

Calling the `/currentUser` endpoint returns a JSON object that provides the user ID and additional information of the logged-on user. The table below describes the properties contained in the JSON object and specifies the principal attribute used to compute this information.

Property Name	Description	Principal Attribute
`name`	The user ID that is used for logging on.	n.a.

Property Name	Description	Principal Attribute
`firstName`	The first name of the user.	firstname
`lastName`	The last name of the user.	lastname
`email`	The email address of the user.	email
`displayName`	Concatenated user name derived from the first name, last name, and user ID. If either the first or the last name does not exist, the `displayName` consists solely of the user ID.	firstname + lastname + name

The `/currentUser` endpoint maps a default set of attributes. To retrieve all attributes, use the `/attributes` endpoint as described in *User Attributes*.

Example

A sample URL for the route defined above would look like this: `/services/userapi/currentUser`.

An example response could return the following user data:

```
{
  "name": "p12345678",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john@doeenterprise.com",
  "displayName": "John Doe (p12345678)"
}
```

Caution

Calls to this service must not be cached by the Content Delivery Network(CDN). Caching causes the wrong results to be returned.

User Attributes

The `/attributes` endpoint returns the principal attributes of the current user as a JSON object. These attributes are received as SAML assertion attributes when the user logs on. To make them visible, define a mapping within the trust settings of the SAP BTP cockpit, see Configure User Attribute Mappings [page 1727].

Example

A sample URL for the route defined above would look like this: `/services/userapi/attributes`.

If the principal attributes `firstname`, `lastname`, `companyname`, and `organization` are present, an example response may return the following user data:

```
{
  "firstname": "John",
```

```
  "lastname": "Doe",
  "companyname": "Doe Enterprise",
  "organization": "Customer sales and marketing"
}
```

Query Parameters

For some endpoints, you can use query parameters to influence the output behavior of the endpoint. The following table shows which parameters exist for the `/attributes` endpoint and how they impact the outputs.

URL Parameter	Type/Unit	Default	Recommended Value	Behavior
`multiValuesAsArrays`	Boolean	`false`	**`true`**	If set to **`true`**, multivalued attributes are formatted as JSON arrays. If set to **`false`**, only the first value of the entire value range of the specific attribute is returned and formatted as a simple string. **i Note** If set to **`true`** for an attribute that is not multivalued, then the value of the attribute is formatted as a simple string and not a JSON array.

4.5.2.5.8 Welcome File

You can either display the default Welcome file or specify a different file as Welcome file.

If the application is accessed only with the domain name in the URL, that is without any additional path information, then the `index.html` file that is located in the root folder of your repository is delivered by default. If you want to deliver a different file, configure this file in the `neo-app.json` file using the `WelcomeFile` parameter. With this additional parameter you specify whether a redirect is sent to the Welcome file or whether the Welcome file is delivered without redirect. If this option is set, then instead of serving the Welcome file directly under `/`, the HTML5 application will send a redirect to the `WelcomeFile` location. With that, relative links in a Welcome file that is not located in the root directory will work.

To configure the Welcome file, add a JSON string with the following format to the `neo-app.json` file:

```
"welcomeFile": "<path to the welcome file>",
"sendWelcomeFileRedirect": true | false
```

Example

An example configuration, which forwards requests without any path information to an `index.html` file in the `/resources` folder would look like this:

```
"welcomeFile": "/resources/index.html",
"sendWelcomeFileRedirect": true
```

4.5.2.5.9 Logout Page

To trigger a logout of the logged-in user, you can configure a logout page in the application descriptor.

When executing a request to the configured logout page, the server triggers a logout. This results in a response containing a logout request that is send to the identity provider (IdP) to invalidate the user's session on the IdP. After the user is logged out from the IdP, the configured logout page is called again. Now, the content of the logout page is served. The logout page is always unprotected, independent of the authentication method of the application and independent of additional security constraints. In case additional resources, for example, SAPUI5, are referenced from the logout page, those resources have to be unprotected as well.

For information on how to configure certain paths as unprotected, see Authentication [page 1130] and Authorization [page 1131].

Because non-active application versions always require authentication, a logout is only triggered for the active application version. For non-active application versions the logout page is served without triggering a logout.

To configure a logout page for your application, use the following format in the `neo-app.json` file:

```
...
    "logoutPage": "<path to logout page>"
...
```

Example

An example configuration of a logout page looks like this:

```
...
    "logoutPage": "/logout.html"
...
```

4.5.2.5.10 Cache Control

To improve the performance of your application you can control the `Cache-Control` headers, which are returned together with the static resource of your application.

You can configure caching for the complete application, for dedicated paths, or resources of the application. If the path you specify ends with a slash character (`/`) all resources in the given directory and its sub-directories are matched. You can also specify the path using wildcards, for example, the path **`**.html`** matches all resources ending with `.html`. Only the first caching directive that matches an incoming request is applied. The path **`**.css`** hides, for example, other paths such as `/resources/custom.css`.

With the `directive` property, you specify whether public proxies can cache the resources. The possible values for the `directive` property are:

- `public`
  The resource can be cached regardless of your response headers.
- `private`
  Your resource is stored by end-user caches, for example, the browser's internal cache only.
- `none`
  This is the default value that does not send an additional directive

To configure caching, add a JSON string in the following format to the `neo-app.json` file:

```
...
"cacheControl": [
                  {
                    "path": "<optional path of resources to be cached>",
                    "directive": "none | public | private",
                    "maxAge": <lifetime in seconds>
                  }
                ]
...
```

Example

An example configuration that caches all static resources for 24 hours looks like this:

```
...
"cacheControl": [
                  {
                    "maxAge": 86400
                  }
                ]
...
```

4.5.2.5.11 Approving HTTP Headers

For security reasons not all HTTP headers are forwarded from the application to a backend or from the backend to the application.

Header Approved by Default

The following HTTP headers are forwarded automatically without any additional configuration because they are part of the HTTP standard:

- `Accept`
- `Accept-Charset`
- `Accept-Language`
- `Accept-Range`
- `Age`

- `Allow`
- `Authorization`
- `Cache-Control`
- `Content-Language`
- `Content-Location`
- `Content-Range`
- `Content-Type`
- `Date`
- `ETag`
- `Expires`
- `From`
- `If-Match`
- `If-Modified-Since`
- `If-None-Match`
- `If-Range`
- `If-Unmodified-Since`
- `Last-Modified`
- `Link`
- `Location`
- `Max-Forwards`
- `Pragma`
- `Range`
- `Referer`
- `Retry-After`
- `User-Agent`
- `Vary`
- `Via`
- `Warning`
- `WWW-Authenticate`

Additionally the following HTTP headers are transferred automatically because they are frequently used by Web applications and (SAP) servers:

- `Content-Disposition`
- `Content-MD5`
- `DataServiceVersion`
- `DNT`
- `MaxDataServiceVersion`
- `Origin`
- `RequestID`
- `Sap-ContextId`
- `Sap-Message`
- `Sap-Messages`
- `Sap-Metadata-Last-Modified`

- `SAP-PASSPORT`
- `Slug`: For more information, see Atom Publishing Protocol .
- `X-CorrelationID`
- `X-CSRF-TOKEN`
- `X-dynaTrace`
- `X-Forwarded-For`
- `X-HTTP-Method`
- `X-Requested-With`

Approving Custom Headers

If you need additional HTTP headers to be forwarded to or from a backend request or backend response, add the header names in the following format to the `neo-app.json` file:

```
"headerWhiteList": [<header1> (, <header2>, ...)]
```

> ### Example
>
> An example configuration that forwards the additional headers `X-Custom1` and `X-Custom2` looks like this:
>
> ```
> "headerWhiteList": ["X-Custom1 "," X-Custom2"]
> ```

Excluded Headers

The following headers are never forwarded:

- `Cookie`
- `Cookie2`
- `Content-Length`
- `Accept-Encoding`

Cookies are used for user session identification and therefore should not be shared. The system stores cookies sent by a backend in the session and removes them from the response before forwarding to the user. With the next request to the backend the stored cookies are added again.

The `Content-Length` header cannot be approved as the value is recalculated on demand matching the content of the given request or response.

The HTML5 application internally manages the `Accept-Encoding` header.

4.5.2.5.12 Custom Response Headers

You can add custom response headers to your application.

Custom response headers are added to an application, for example, to comply with security standards.

To set default HTTP response headers retrieved by the request, add the header names and values in the following format to the `neo-app.json` file:

> **i Note**
>
> If back end requests are retrieved with the same response headers as in the neo-app.json file, then those values are not overridden.

> **i Note**
>
> Custom Response headers are not supported when running your application from SAP Web IDE.

```
"responseHeaders": [
        {
            "headers": [
                {
            "name": "header name",
                    "value": "header value"
                }
            ]
        }
    ],
```

> **Sample Code**

```
 "responseHeaders": [
         {
             "headers": [
                 {
             "name": "Content-Security-Policy",
                     "value": "default-src 'self'"
                 }
             ]
         }
     ],
```

> **i Note**
>
> The Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware.

4.6 API Documentation

API documentation for the Neo environment.

SAP HANA XS JavaScript Reference

SAP BTPSDK for iOS

REST APIs

Lifecycle Management

Java ALM API

Reverse Proxy

SAP Virtual Machine Service API (Basic Authentication)

SAP Virtual Machine Service API (OAuth)

SAP Audit Log Service

Audit Log Retrieval API for the Neo Environment

Audit Log Retention API for the Neo Environment

Audit Log Retrieval API for the Cloud Foundry Environment

SAP Monitoring Service

Alerting Channels API

Checks API

Metrics API

Security

Authorization Management API

Platform Authorization Management API

Reverse Proxy API

Password Storage API

Keystore API

More APIs

SAP API Business Hub

Java API Documentation for Neo Environment

The Java API documentation for the Neo environment is provided as part of the downloadable SDK archives. To get to it, do the following:

1. Install the SDK for Neo environment of your choice. See Install the SAP BTP SDK for Neo Environment [page 845].

2. On your local machine, navigate to the folder of the archive you downloaded and extracted.
3. Navigate to the *javadoc* folder, and open *index.html* in your Web browser.

Related Information

Java ALM API Tutorial [page 877]
Audit Log Retrieval API Usage for the Neo Environment [page 1810]
Audit Log Retention API Usage for the Neo Environment [page 1815]
Audit Log Retrieval API Usage for the Cloud Foundry Environment
Get Started: SAP Virtual Machine Service API (Neo Environment)
Consuming the Destination Configuration Service [page 230]

4.6.1 Using Platform APIs

Platform APIs are protected with *OAuth 2.0 client credentials*. Create an OAuth client and obtain an access token to call the platform API methods.

Context

For description of OAuth 2.0 client credentials grant, see the OAuth 2.0 client credentials grant specification .

For some of the platform APIs, the JWT type of access tokens is used. See Best Practices for Resilient OAuth 2.0 Communication [page 1768].

For detailed description of the available methods, see the respective API documentation.

> → Tip
>
> Do not get a new OAuth access token for each and every platform API call. Re-use the same existing access token throughout its validity period instead, until you get a response indicating the access token needs to be re-issued.

1. Create an OAuth Client

Context

The OAuth client is identified by a *client ID* and protected with a *client secret*. In a later step, those are used to obtain the OAuth API access token from the *OAuth access token endpoint*.

Procedure

1. In your Web browser, open the SAP BTP cockpit.
2. Go to the *Security* > *OAuth* section.
3. Enter the *Platform API* tab.
4. Choose *Create API Client*.
5. Choose the required API option and required combination of scopes. We recommend that you enter an OAuth client description as well.
6. Save the client.
7. Copy the generated client ID and client secret, and save them locally.

> ⚠ Caution
>
> Make sure you save the generated client credentials. Once you close the confirmation dialog, you cannot retrieve the generated client credentials again.

2. Get an OAuth Access Token

Context

OAuth access token endpoint and use the *client ID* and *client secret* as user and password for HTTP Basic Authentication. You will receive the access token as a response.

By default, the access token received in this way is valid 1500 seconds (25 minutes). You cannot configure its validity length.

If you want to revoke the access token before its validity ends, delete the respective OAuth client. The access token remains valid up to 2 minutes after the client is deleted.

Procedure

1. Send a POST request to the OAuth access token endpoint. The URL is landscape specific, and looks like this:

   https://api.<cloud platform host>/oauth2/apitoken/v1?grant_type=client_credentials

   See Regions.

   The parameter *grant_type=client_credentials* notifies the endpoint that the Client Credentials flow is used.
2. Get and save the access token from the received response from the endpoint.

   The response is a JSON object, whose *access_token* parameter is the access token. It is valid for the specified time (in seconds) in the *expires_in* parameter. (default value: 1500 seconds).

Example

Retrieving an access token on the trial landscape will look like this:

```
POST https://api.hanatrial.ondemand.com/oauth2/apitoken/v1?
grant_type=client_credentials
Headers:
Authorization: Basic eW91ckNsaWVudElEOnlvdXJDbGllbnRTZWNyZXQ
```

The `eW91ckNsaWVudElEOnlvdXJDbGllbnRTZWNyZXQ` String in the above request is the Base-64 encoded `<clientID>:<ClientSecret>`.

You receive a response like this:

```
{
    "access_token": "51ddd94b15ec85b4d54315b5546abf93",
    "token_type": "Bearer",
    "expires_in": 1500,
    "scope": "hcp.manageAuthorizationSettings hcp.readAuthorizationSettings"
}
```

Alternatively, you can do this using an HTTP destination.

1. At the required (application, subaccount or global account) level, create an HTTP destination with the following information (the name can be different):
   - Name=<yourdestination name>
   - URL=https://api.<cloud platform host>/oauth2/apitoken/v1?grant_type=client_credentials
   - ProxyType=Internet
   - Type=HTTP
   - CloudConnectorVersion=2
   - Authentication=BasicAuthentication
   - User=<clientID>
   - Password=<clientSecret>

   See Create HTTP Destinations [page 80].
2. In your application, obtain an `HttpURLConnection` object that uses the destination.
   See ConnectivityConfiguration API [page 139].
3. With the object retrieved from the previous step, execute a POST call.

   ```
   urlConnection.setRequestMethod("POST");
   urlConnection.setRequestProperty("Authorization", "Basic <Base64 encoded
   representation of {clientId}:{clientSecret}>");
   urlConnection.connect();
   ```

3. Call API Methods

Procedure

In the requests to the required platform API, include the access token as a header with name *Authorization* and value *Bearer <token value>*.

Example

We will call the Authorization Management API.

```
GET https://api.hanatrial.ondemand.com/authorization/v1/accounts/p1234567trial/
users/roles/?userId=myUser
Headers:
Authorization: Bearer 51ddd94b15ec85b4d54315b5546abf93
```

Related Information

Keystore API [page 1779]
Authorization Management API [page 1687]
Audit Log Retrieval API Usage for the Neo Environment [page 1810]
Audit Log Retention API Usage for the Neo Environment [page 1815]
Platform Authorization Management API [page 1688]

4.7 Using Services in the Neo Environment

Learn more about using services in the Neo environment:

- About Services in the Neo Environment [page 1152]
- Enable Services in the Neo Environment [page 1153]
- Disable Services in the Neo Environment [page 1154]

4.7.1 About Services in the Neo Environment

In the Neo environment, you enable services in the SAP BTP cockpit.

The cockpit lists all services grouped by service category. Some of the services are basic services, which are provided with SAP BTP and are ready-to-use. In addition, extended services are available. A label on the tile for a service indicates if this service is enabled.

An administrator must first enable the service and apply the service-specific configuration (for example, configure the corresponding roles and destinations) before any subaccount members can use it.

> **i Note**
>
> Some services are exposed only for trial accounts. That means the services are not, or not yet, released for use with a customer or partner account.
>
> Some services are exposed only if your organization has purchased a license.

The *Service Description* section offers the following:

- A short description for the selected service
- (Optional) Links to documentation for the selected service
- (Optional) A link to the service UI. Some services provide a UI while others are consumable only via API.

The *Service Configuration* section offers the following:

- (Optional) Links to configuration screens of the service.

> → Remember
>
> You can access most of the links only after the service has been enabled.

The configuration options for a service may look like in this example for the Portal service:

- To configure connection parameters to other systems (by creating connectivity destinations), choose *Configure* `<Portal Service>` *Destinations*.
  This option is available only if the service is enabled.
- To create custom roles and assign custom or predefined roles to individual users and groups, choose *Configure* `<Portal Service>` *Roles*.
  This option is available only if the service is enabled.

4.7.2 Enable Services in the Neo Environment

In the Neo environment, you might need to enable services before subaccount members can integrate them with applications. Note that free services are always enabled.

Prerequisites

The Administrator role for the subaccount.

Context

Procedure

1. Navigate to the subaccount in which you'd like to enable a service. For more information, see Navigate in the Cockpit.
2. In the navigation area, choose *Services*.
3. Select the service and choose *Enable*.

4.7.3 Disable Services in the Neo Environment

In the Neo environment, you might need to disable services so that they are not available to subaccount members.

Prerequisites

The Administrator role for the subaccount.

Context

Procedure

1. Navigate to the subaccount in which you'd like to disable a service. For more information, see Navigate in the Cockpit.
2. In the navigation area, choose *Services*.
3. Select the service and choose *Disable*.

   All data generated by the service is deleted and cannot be retrieved.

   > **i** Note
   >
   > - If other services use the service, they may be negatively impacted when you disable it. Your service documentation may provide information about services that are dependent on your service.
   > - Services in the Neo environment generally have a delay period before they can be reenabled in the subaccount. In these cases, the confirmation message displays the number of hours before you can reenable the service.
   > - The services enabled by default in the Neo environment cannot be disabled. Such services do not cause performance impact as long as they are not used.
   > - If the *Disable* button is dimmed, it means that you'll not be able to disable it yourself. You'll need to report an incident to SAP Support to assist you with disabling the service.

5 Extensions, Neo Environment

The extension capabilities of SAP Business Technology Platform (SAP BTP) enables developers to implement loosely coupled extension applications securely, thus implementing additional workflows or modules on top of the existing SAP cloud solution they already have.

Introduction

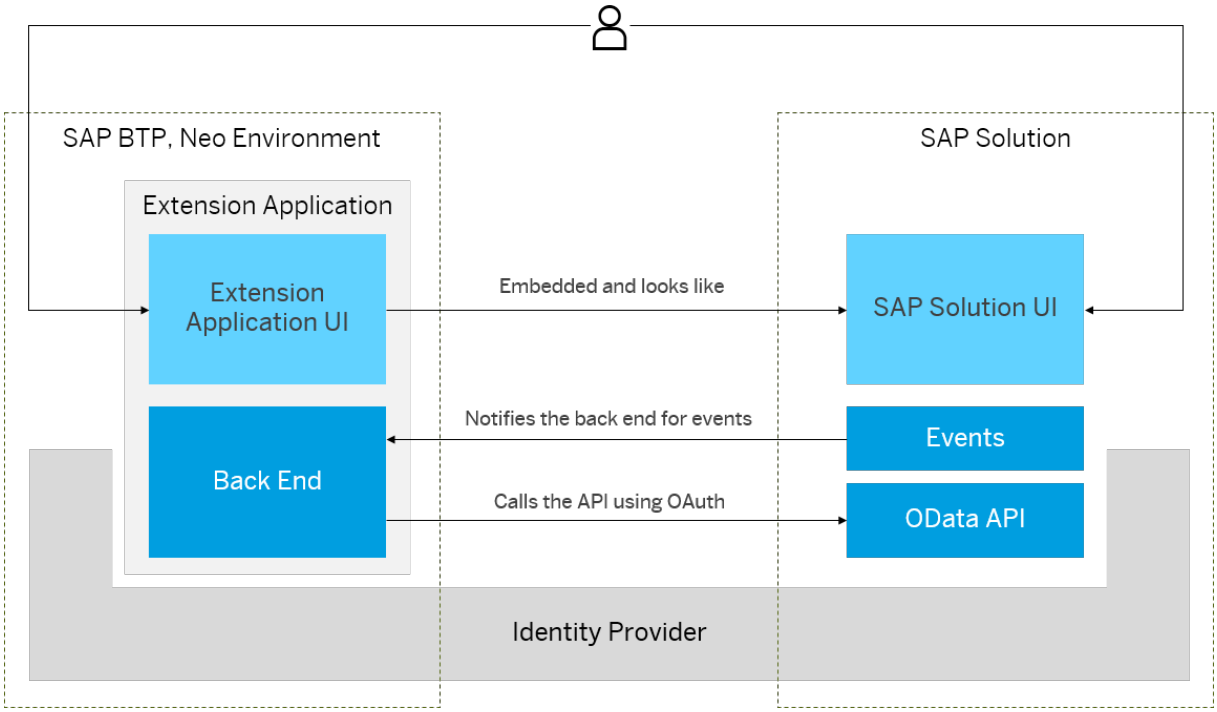
All standard SAP solutions are offered with customizing capabilities. Additionally, customers often have their own requirements for innovative or industry-specific extensions and the extension capability of SAP BTP can help them build, deploy, and operate their new functionalities easily and securely.

Extension Application

SAP BTP, Neo environment offers ready-to-use development and runtime environment in the cloud. You can extend standard SAP solutions without disrupting their performance and core processes. When building extension applications, you can also benefit from the automation of the integration between the SAP BTP, Neo environment and the extended SAP solutions.

These are the specifics of an extension application:

- The extension application needs to call APIs from the SAP solution.
- The SAP solution (for example, SAP SuccessFactors) can notify the extension application using APIs exposed from the extension application.
- The UI of the extension application can:
  - Be embedded in the UI of the SAP solution
  - Be separate from the SAP solution, for example embedded into a corporate or a consumer portal



Extension Subaccount

An extension subaccount is part of a customer or partner global account in SAP BTP which is configured to interact with a particular SAP solution through standardized destinations, usually with identity propagation turned on. This subaccount is paired with the extended SAP solution.

Provide Extension Applications

There are two deployment models for providing an extension application:

- Provider-managed deployment model: an extension application is deployed in a separate subaccount in SAP BTP, and the customer's extension subaccount is subscribed for this application.
  For example, an SAP partner can use this deployment model to offer his extension applications as a software-as-a-service (SaaS).
- Consumer-managed deployment model: an extension application is deployed directly in the extension subaccount of the customer.
  This deployment model can be used for extension applications built by the customer, or purchased from an SAP partner as a deployable solution.

Related Information

Extending SAP SuccessFactors in the Neo Environment [page 1175]
Extending SAP S/4HANA Cloud in the Neo Environment [page 1157]
Extending SAP Cloud for Customer in the Neo Environment [page 1219]
Extending SAP Ariba in the Neo Environment [page 1240]

5.1 Extending SAP S/4HANA Cloud in the Neo Environment

This section guides you through the configuration tasks that you need to perform to enable the SAP BTP, Neo environment for developing extension applications for your SAP S/4HANA Cloud tenant.

Overview

With the extension capabilities of SAP BTP, you can create SAP S/4HANA Cloud side-by-side extensions: they extend the SAP S/4HANA Cloud functionality but reside on the cloud platform.

To do that, you need to own an SAP S/4HANA Cloud tenant. The authentication against the SAP S/4HANA Cloud tenant is based on the Identity Authentication tenant that is provided together with the SAP S/4HANA Cloud tenant. Typically, you configure this Identity Authentication tenant to forward authentication requests to your corporate identity provider.

Prerequisites

- Your company has an enterprise account for the cloud platform.
- You have a user account on the platform with administrative permissions for the enterprise of your company and you can use the SAP BTP cockpit.

Process Flow

You have to configure the cloud platform extension integration with SAP S/4HANA Cloud to enable the use of applications running on top of the platform from SAP S/4HANA Cloud.

These are the steps you need to follow:

1. Set up the subaccount in SAP BTP by configuring the single sign-on (SSO). See Configuring Single Sign-On Between SAP S/4HANA Cloud and SAP BTP, Neo Environment [page 1158]
2. Configure the extension application and create the respective destination in the SAP BTP cockpit. See Configuring the Extension Application [page 1161].

5.1.1 Configuring Single Sign-On Between SAP S/4HANA Cloud and SAP BTP, Neo Environment

You configure the integration between SAP BTP, Neo environment and SAP S/4HANA Cloud to allow the SAP S/4HANA Cloud side-by-side extension applications to run on top of the cloud platform.

Context

Identity Authentication provides authentication, single sign-on (SSO), and on-premise integration. Identity Authentication service is closely integrated with SAP BTP, and it is offered as part of the platform.

To ensure the required security for accessing the applications, you need to configure SSO between the subaccount in SAP BTP and the SAP S/4HANA Cloud tenant using a SAML identity provider, for example Identity Authentication. The SSO requires both solutions to be configured as trusted SAML service providers for the Identity Authentication service, and at the same time, the Identity Authentication service to be configured as trusted identity provider for the two solutions.

> **i Note**
>
> You own an SAP S/4HANA Cloud tenant with an Identity Authentication tenant configured. You need to use the same Identity Authentication tenant for your subaccount in SAP BTP.

Procedure

Configure SSO between the subaccount in SAP BTP and the SAP S/4HANA Cloud tenant using Identity Authentication as an identity provider. For more information, see Identity Authentication Tenant as an Application Identity Provider [page 1734].

Results

The trust will be established automatically upon registration on both the SAP BTP, Neo environment and the tenant for Identity Authentication service side with the following default configuration settings:

- Principal propagation: disabled

  If needed, you can enable it. To do so, choose *Principal Propagation* > *Enable*.
  If enabled, this option enables applications to propagate principal information to each other. Choose this value if you want to enable application-to-application SSO for applications running in the subaccount. Otherwise, set this option to *Disabled*.
- *Login Name* name is not set as a logon identifier for your users. If needed, you can enable it. For more information, see Configure Logon Identifiers.

Alternatively, you can configure SSO between the Identity Authentication tenant and SAP BTP, Neo environment manually. For more information, see Optional: Configure Single-Sign On Manually [page 1159].

> → Tip
>
> We recommend that you use the manual configuration only if the automated configuration is not possible. For example, if you are not authorized to access the Identity Authentication tenant.

5.1.1.1 Optional: Configure Single-Sign On Manually

Use this procedure as an alternative to the automated configuration of the SSO between SAP BTP and the Identity Authentication tenant.

Prerequisites

> → Tip
>
> We recommend that you use the manual configuration only if the automated configuration is not possible. For example, if you are not authorized to access the Identity Authentication tenant.

You own an SAP S/4HANA Cloud tenant with an Identity Authentication tenant configured. You need to use the same Identity Authentication tenant for your subaccount. For more information about how to get Identity Authentication, see Getting Started with Identity Authentication Service.

Context

The Identity Authentication service is closely integrated with SAP BTP, and it is offered as part of most of the cloud platform packages. For those packages the trust between the subaccount and Identity Authentication service is configured automatically and the tenant for the Identity Authentication service is set up by default, once you have a partner or customer subaccount. However, you can manually configure the trust and set up the Identity Authentication tenant if your scenario requires it.

Procedure

1. Open the SAP BTP cockpit and select the region in which your subaccount is hosted. Select the global account that contains your subaccount, and then choose the tile of your subaccount. For more information about regions, see Regions and Hosts.
2. Choose *Security* > *Trust Configuration*.
3. From the *Local Service Provider* tab select *Edit*.
4. Change the *Configuration Type* to *Custom*.
5. From the *Principal Propagation* dropdown box, select *Enabled*. Thus, you enable the propagation of the principal ID in the OAuth destination and hence, the application-to-application single sign-on (SSO).

6. Configure the signing key and the signing certificate as follows:
    - If you want to use a signing key and a self-signed certificate automatically generated by the system, choose *Generate Key Pair*.
    - If you have your own key and certificate generated from an external application and signed by a trusted CA, you can use them instead of using the ones generated by the SAP BTP. To do so, copy the Base64-encoded signing key in the *Signing Key* field, and then copy the textual content of the certificate in the *Signing Certificate* field.
7. Choose *Save*.
8. Choose the *Get Metadata* link to download the SAP BTP metadata for your subaccount. You will need this metadata in **Step 13**.
9. Access the administration console of the Identity Authentication tenant, using the following URL:

    `https://<tenant ID>.accounts.ondemand.com/admin`

    You can also get the URL from the Identity Authentication tenant registration e-mail.

    > **i Note**
    >
    > You need to use another browser, or incognito session of the same browser.

10. Choose *Applications & Resources* > *Applications*.
11. Choose the *+Add* button on the left-hand panel, and enter the name of your subaccount.
12. Choose *Save*.
13. Configure the SAML 2.0 trust with the subaccount as a service provider. To do so, proceed as follows:
    a. On the left-hand side, choose the newly created application, and then choose *Trust*.
    b. Choose *SAML 2.0 Configuration*.
    c. Upload the metadata XML file of your subaccount that you have downloaded in **Step 8**.

        On service provider metadata upload, the fields are populated with the parsed data from the XML file.
    d. Save the configuration settings.
14. Configure the identity federation on the Identity Authentication service. To do so, proceed as follows:
    a. You are still in the tenant's administration console for the Identity Authentication service. Under *Applications and Resources*, choose the *Tenant Settings* tile, and then select *Login Name*.

        This is the profile attribute that the Identity Authentication service sends to the application as a name ID. The application then uses this attribute to identify the user.
    b. Save your selection.
15. Under *CONDITIONAL AUTHENTICATION* choose *Conditional Authentication*.
16. From the drop-down menu in the *Default Authenticating Identity Provider* section, select your identity provider
17. Choose *Save*.
18. You have to save the metadata of your Identity Authentication tenant on your local file system as an XML file. You can either find the tenant at `https://<tenant ID>.accounts.ondemand.com/saml2/metadata` or access it via *Applications & Resources* > *Tenant Settings* > *SAML 2.0 Configuration*. Then choose the *Download Metadata File* link. You will need this metadata in **Step 21**.
19. Go back to your subaccount in the SAP BTP cockpit and choose *Security* > *Trust*.
20. Select the *Application Identity Provider* tab.

21. Select *Add Trusted Identity Provider*.
22. In the *General* tab, upload the Identity Authentication service metadata XML file (from **Step 17**) in the *Metadata File* field.
23. Choose *Save*.

Results

The trust will be established automatically upon registration on both the SAP BTP and Identity Authentication tenant side.

Related Information

Get Started with Identity Authentication
ID Federation with a Identity Authentication Tenant

5.1.2 Configuring the Extension Application

To set up the extension application, you have to configure the connectivity from the subaccount in SAP BTP to the SAP S/4HANA Cloud tenant.

When configuring the extension application, you have to set up the connectivity from the subaccount in SAP BTP where you will deploy this extension application to the SAP S/4HANA Cloud tenant. To do that, you can use different authentication methods: Basic, Client Certificate, or SAML Bearer Assertion.

See Configure the Extension Applications's Connectivity to SAP S/4HANA Cloud [page 1161].

5.1.2.1 Configure the Extension Applications's Connectivity to SAP S/4HANA Cloud

To set up the connectivity from a subaccount in SAP BTP to an SAP S/4HANA Cloud tenant, you need to create HTTP destinations in the SAP BTP cockpit. These destinations provide data communication via HTTP protocol. For more information, see HTTP Destinations.

When creating an HTTP destination, you can use different authentication types for access control:

Basic Authentication

Using authentication method you need to provide username and password. When configuring the HTTP destination in the cockpit, the username must correspond to the communication user in the SAP S/4HANA Cloud tenant.

See Using Basic Authentication [page 1162].

Client Certificate Authentication

To configure a client certificate authentication, you need a client certificate signed by a trusted certificate authority (CA). You upload the public key when creating a communication user in the SAP S/4HANA Cloud tenant, and then, you add the corresponding keystore to the HTTP destination in the cockpit.

See Using Client Certificate Authentication [page 1165].

SAML Bearer Assertion Authentication

You can use the SAML Bearer assertion flow for consuming OAuth-protected resources. Users are authenticated by using SAML against the configured trusted identity providers. The SAML assertion is then used to request an access token from an OAuth authorization server. This access token is automatically injected in all HTTP requests to the OAuth-protected resources.

See Using SAML Bearer Assertion Authentication [page 1167].

5.1.2.1.1 Using Basic Authentication

Context

To be able to use Basic authentication, you need to configure both SAP S/4HANA Cloud and SAP BTP sides.

Related Information

Set Up SAP S/4 HANA Cloud Side [page 1163]

Set Up SAP BTP Side [page 1164]

5.1.2.1.1.1 Set Up SAP S/4 HANA Cloud Side

Context

From the SAP S/4HANA Cloud side you need to maintain the communication settings to configure the connectivity between SAP S/4HANA Cloud and SAP BTP.

Procedure

1. Create a communication user. See Maintain Communication Users.
2. Create a communication system. See Maintain Communication Systems.
   a. Log into the SAP Fiori launchpad in the SAP S/4HANA Cloud system.
   b. Select the *Communication Systems* tile.
   c. Choose *New* to create a new system.
   d. Enter a system ID and a system name.
   e. Choose *Create*.
   f. Enter information about SAP BTP in the *Technical Data* section.
   g. Choose + (Add) under *User for Inbound Communication*.
   h. In the dialog box that appears, select *User Name and Password* from the *Authentication Method* drop-down list.

      The username corresponds to the communication user.
   i. Choose *OK* to confirm.
   j. Choose *Save*.
3. Create a communication arrangement and select a communication scenario. See Maintain Communication Arrangements.

Related Information

Communication Management

5.1.2.1.1.2 Set Up SAP BTP Side

Prerequisites

You have logged into the SAP BTP cockpit from the landing page for your subaccount.

Context

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose *Connectivity* > *Destinations* in the navigation panel.
3. Create an HTTP destination as follows:

   Create an *BasicAuthentication* HTTP destination and configure its settings as follows:

Parameter	Value
`Name`	Enter a meaningful name.
`Type`	**`HTTP`**
`Description`	(Optional) Enter a meaningful description.
`URL`	The service URL from the communication arrangement.
`Proxy Type`	**`Internet`**
`Authentication`	**`BasicAuthentication`**
`User`	The name of the communication user you have in the SAP S/4HANA Cloud tenant.
`Password`	The password for the communication user.

4. Select the *Use default JDK truststore* checkbox.
5. Save your entries.

5.1.2.1.2 Using Client Certificate Authentication

Context

To be able to use client certificate authentication, you need to configure both SAP S/4HANA Cloud and SAP BTP sides.

Related Information

Set Up SAP S/4HANA Cloud Side [page 1165]
Set Up SAP BTP Side [page 1166]

5.1.2.1.2.1 Set Up SAP S/4HANA Cloud Side

Context

To use client certificate authentication, you first start with creating and configuring the communication settings in the SAP S/4HANA Cloud tenant. To do that, you have to:

Procedure

1. Obtain a client certificate signed by a trusted certificate authority (CA) in *.pem* format. See Keys and Certificates.

   You can find a list of the trusted CA in the SAP S/4HANA Cloud tenant using the **Maintain Certificate Trust List** application. See Maintain Certificate Trust List.
2. Create a communication user and upload the public key. See Maintain Communication Users.
3. Create a communication system and add the communication user as *User for Inbound Communication* with an authentication method SSL Client Certificate.
   a. Log into the SAP Fiori launchpad in the SAP S/4HANA Cloud system.
   b. Select the *Communication Systems* tile.
   c. Choose *New* to create a new system.
   d. Enter a system ID and a system name.
   e. Choose *Create*.

f. Enter information about SAP BTP in the *Technical Data* section.
g. Choose + (Add) under *User for Inbound Communication*.
h. In the dialog box that appears, select *SSL Client Certificate* from the *Authentication Method* drop-down list.

   The username corresponds to the communication user.
i. Choose *OK* to confirm.
j. Choose *Save*.

4. Create a communication arrangement using an existing or create a new scenario that supports client certificate authentication. See Maintain Communication Arrangements.

   You can use the already created communication system. The settings in the *Inbound Communication* section are filled in automatically. Save the value from the *URL* field, you will need it when creating a destination in the subaccount in SAP BTP.

5.1.2.1.2.2 Set Up SAP BTP Side

Prerequisites

You have logged into the SAP BTP cockpit from the landing page for your subaccount.

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose *Connectivity* > *Destinations* in the navigation panel.
3. Create an HTTP destination as follows:

   To enable the support of SSO, create an *ClientCertificateAuthentication* HTTP destination and configure its settings as follows:

Parameter	Value
`Name`	Enter a meaningful name.
`Type`	**`HTTP`**
`Description`	(Optional) Enter a meaningful description.
`URL`	The service URL from the communication arrangement.
`Proxy Type`	**`Internet`**

Parameter	Value
`Authentication`	**`ClientCertificateAuthentication`**

4. Choose *Upload and Delete Certificate* link to upload your keystore. The keystore format .jks. When you finish uploading, choose Close.
   a. From the *Key Store Location* drop-down menu, select your keystore.
   b. In the *Key Store Password*, enter the keystore password.
5. Select the *Use default JDK truststore* checkbox.
6. Save your entries.

5.1.2.1.3 Using SAML Bearer Assertion Authentication

Context

To be able to use SAML Bearer Assertion authentication, you need to configure both SAP S/4HANA Cloud and SAP BTP sides.

Related Information

SAML Bearer Assertion Authentication
Set Up SAP S/4HANA Cloud Side [page 1167]
Set Up SAP BTP Side [page 1169]

5.1.2.1.3.1 Set Up SAP S/4HANA Cloud Side

Context

From the SAP S/4HANA Cloud side you need to maintain the communication settings to configure the connectivity between SAP S/4HANA Cloud and SAP BTP.

> **i** Note
>
> Make sure you have assigned the appropriate business catalogs to the propagated business users in the SAP S/4HANA Cloud tenant.

Procedure

1. Create a communication user. See Maintain Communication Users.
2. Create a communication system. See Maintain Communication Systems.
   a. Log into the SAP Fiori launchpad in the SAP S/4HANA Cloud system.
   b. Select the *Communication Systems* tile.
   c. Choose *New* to create a new system.
   d. Enter a system ID and a system name.
   e. Choose *Create*.
   f. In the *Technical Data* section, in the *Host Name* field, enter the host of the subaccount in SAP BTP. To find all available hosts in the Neo environment, see Regions and Hosts Available for the Neo Environment.
   g. Enable the OAuth Identity Provider by checking the box under *OAuth 2.0 Identity Provider*.
   h. Upload the signing certificate. This certificate is created from the metadata file you downloaded from your subaccount in SAP BTP in Optional: Configure Single-Sign On Manually [page 1159], Step 8. Follow these steps to prepare the certificate and upload it.
      1. Open the `metadata` file with an editor of your choice. For the next configuration steps you will need the value of the `X509Certificate` tag.
      2. Create a new file and copy the content between the `<X509Certificate>` tags into this file.
      3. Add `-----BEGIN CERTIFICATE-----` to the top and `-----END CERTIFICATE-----` to the end of the file.
      4. Save as a `x509.cer` file.
      5. Back in the *Communication Systems* application select *Upload Signing Certificate* and upload the certificate you created.
   i. Choose + (Add) under *User for Inbound Communication*.
   j. In the dialog box that appears, select *User Name and Password* from the *Authentication Method* drop-down list.

      The username corresponds to the communication user.
   k. Choose *OK* to confirm.
   l. Choose *Save*.
3. Create a communication arrangement and select a communication scenario. See Maintain Communication Arrangements.

> **i Note**
>
> When you have the communication arrangement created, choose *OAuth 2.0 Details*. Copy and save locally the fields and their values. You will need them when setting up the destination in the SAP BTP cockpit.

5.1.2.1.3.2 Set Up SAP BTP Side

Prerequisites

You have logged into the SAP BTP cockpit from the landing page for your subaccount.

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose *Connectivity* > *Destinations* in the navigation panel.
3. Create an HTTP destination as follows:

   To enable principal propagation, create an *OAuth2SAMLBearerAssertion* HTTP destination and configure its settings as follows:

   1. Configure the basic settings:

Parameter	Value
`Name`	Enter a meaningful name.
`Type`	**`HTTP`**
`Description`	(Optional) Enter a meaningful description.
`URL`	The service URL from the communication arrangement.
`Proxy Type`	**`Internet`**
`Authentication`	**`OAuth2SAMLBearerAssertion`**
`Audience`	This is the *SAML2 Audience* from the *OAuth 2.0 Details* in the communication arrangement. See Set Up SAP S/4HANA Cloud Side [page 1167], **step 3**.
`Client Key`	The name of the communication user you have in the SAP S/4HANA Cloud tenant.
`Token Service URL`	This is the *Token Service URL* from the *OAuth 2.0 Details* in the communication arrangement. See Set Up SAP S/4HANA Cloud Side [page 1167], **step 3**.
`Token Service User`	The name of the communication user you have in the SAP S/4HANA Cloud tenant.

Parameter	Value
`Token Service Password`	The password for the communication user.
`System User`	This parameter is not used, leave the field empty.

2. Configure the required additional property. To do so, in the *Additional Properties* panel, choose *New Property*, and enter the following property:

Parameter	Value
`authnContextClassRef`	**`urn:oasis:names:tc:SAML:2.0:ac:classes:X509`**
scope	(Optional) This parameter restricts the APIs that can be used by the OAuth Client. ○ If the scope parameter is omitted, then all APIs that are exposed for the OAuth Client can be used. ○ If the scope parameter is maintained, then only the related APIS can be used by the OAuth Client. The value of the OAuth 2.0 scope parameter expressed as a list of space-delimited, case-sensitive strings.

3. Select the *Use default JDK truststore* checkbox.

4. Save your entries.

5.1.3 Configuring SAP Web IDE

Context

When developing a UI for an SAP S/4HANA Cloud solution, you can:

- Test the UI in the SAP Web IDE. In this case, the particular service must be exposed via a communication arrangement.
- Deploy the UI application developed in the SAP Web IDE in your SAP S/4HANA Cloud solution. In this case, a communication arrangement needs to be created for the SAP_COM_0013 scenario.

Within your account in SAP BTP, you are automatically subscribed to the SAP Web IDE. The authentication against the SAP Web IDE is based on the Identity Authentication tenant that is provided together with the SAP S/4HANA Cloud tenant.

In this scenario these are the important aspects:

- The single sign-on configuration between SAP S/4HANA Cloud and SAP BTP ensures the secure and consistent data access for the applications.

- OAuth configuration for OData connectivity for enabling the use of SAP S/4HANA OData APIs ensures the secure data communication with the extended solution.
- With the SAP S/4HANA Cloud tenant provisioning by default it is established a trust between the Identity Authentication tenant and the SAP S/4HANA Cloud tenant and the following business catalogs assigned:

Business Catalog ID	Area	Description
SAP_BCR_CORE_COM	Communication Management	Users assigning communication arrangements to the key user
SAP_BCR_CORE_EXT_UI	Extensibility – Deployment of Fiori Applications	Users creating UI extensions
SAP_BCR_CORE_EXT	Extensibility	Users creating extensions **i Note** You need this catalog if you want to create custom business objects.
SAP_CORE_BC_EXT_TST	Extensibility – Custom Apps and Services	You need this catalog if you want to implement and test a UI against a custom business object.

Procedure

1. Configure the Single Sign-On (SSO).

   A single sign-on (SSO) configuration between SAP S/4HANA Cloud and SAP BTP and the principal propagation enablement ensure the secure and consistent access for the extension solutions. You need to use SSO and OData access with principal propagation, to ensure that the data is accessed on behalf of the proper authorized user.

   Follow the steps described in Optional: Configure Single-Sign On Manually [page 1159].

   1. Configure the SAP BTP trust.
   2. Create a certificate and get the metadata file. Then, import it in the Identity Authentication tenant.
2. Set up the extension application.

   To set up the extension application, you need to configure the connectivity between SAP BTP and SAP S/4HANA Cloud tenant.

   The connectivity configuration steps are required on both systems:

   i Note

   The OAuth authentication method can be used only when a common identity provider is configured for both systems.

1. On SAP S/4HANA Cloud side, you have to maintain the communication settings. See **step 1** from Connectivity Settings Using SAML Bearer Assertion Authentication [page 1173].
2. On the SAP BTP side, you have to create an HTTP destination with *OAuth2SAMLBearerAssertion* for authentication method. See **step 2** from Connectivity Settings Using SAML Bearer Assertion Authentication [page 1173].

3. Check the setup of the OAuth communication.

   You need to check whether the whole setup is working. See Check the Setup of the OAuth Communication [page 1174].

5.1.3.1 Configure and Check the SSO Configuration

You can check that the SSO is properly configured by launching SAP Web IDE.

Context

Procedure

1. Configure the SSO. See Configuring Single Sign-On Between SAP S/4HANA Cloud and SAP BTP, Neo Environment [page 1158].
2. Check if you have set up the SSO correctly.
   a. In the SAP BTP cockpit, go to *Services* and search for **`Web IDE`**.
   b. Choose the *SAP Web IDE Full-Stack* tile.
   c. Choose the *Configure Service* link.
   d. Set up the *DiDeveloper* role. See Assign Users Permission for SAP Web IDE.
   e. Open the link in the *Application URL* field to launch the SAP Web IDE.

Results

You will see the login screen of the Identity Authentication. You need to use another browser, or an incognito session of user.

5.1.3.2 Connectivity Settings Using SAML Bearer Assertion Authentication

Context

To set up the SAP Web IDE, you need to configure the connectivity between SAP BTP and SAP S/4HANA Cloud tenant. You have to use SAML Bearer Assertion authentication, and you need to configure both SAP S/4HANA Cloud and SAP BTP sides.

The general procedure is described in Using SAML Bearer Assertion Authentication [page 1167]. These are the SAP Web IDE specifics you need to take into account:

Procedure

1. Set Up SAP S/4HANA Cloud Side [page 1167].
   a. Create a communication user.
   b. Create a communication system.

      In step 2.f, when specifying the *Technical Data* section, you need to specify a hostname. The hostname of the SAP Web IDE is the SAP Web IDE URL. You can get it by following the steps in Configure and Check the SSO Configuration [page 1172].

      > **i Note**
      >
      > If you have integrated SAP Web IDE with SAP S/4HANA Cloud 1805 or earlier using the communication scenario SAP_COM_0013, you need to update the integration. See
      >
      > Updating the Integration of SAP S/4HANA and SAP Web IDE.

   c. Create a communication arrangement and select a communication scenario.

      In this case the scenario is `SAP_COM_0013`, SAP Web IDE Integration.
2. Set Up SAP BTP Side [page 1169].

   When creating an HTTP destination, these are the SAP Web IDE-specific values:

   a. Configure the basic settings:

Parameter	Value
`Name`	Enter a meaningful name.
`Type`	**`HTTP`**
`Description`	(Optional) Enter a meaningful description.

Parameter	Value
URL	The service URL from the communication arrangement. To do that, copy the first part of the *Service URL/Service Interface* column (the part before /sap/..).
Proxy Type	**Internet**
Authentication	**OAuth2SAMLBearerAssertion**
Audience	This is the *SAML2 Audience* from the *OAuth 2.0 Details* in the communication arrangement. See Set Up SAP S/4HANA Cloud Side [page 1167], **step 3**.
Client Key	The name of the communication user you have in the SAP S/4HANA Cloud tenant.
Token Service URL	This is the *Token Service URL* from the *OAuth 2.0 Details* in the communication arrangement. See Set Up SAP S/4HANA Cloud Side [page 1167], **step 3**.
Token Service User	The name of the communication user you have in the SAP S/4HANA Cloud tenant.
Token Service Password	The password for the communication user.
System User	This parameter is not used, leave the field empty.

b. Configure the required additional properties. To do so, in the *Additional Properties* panel, choose *New Property*, and enter the following property:

Parameter	Value
authnContextClassRef	**urn:oasis:names:tc:SAML:2.0:ac:classes:X509**
WEBIDEEnabled	**true**
WEBIDEUsage	**odata_abap,ui5_execute_abap,dev_abap**

5.1.3.3 Check the Setup of the OAuth Communication

Context

Procedure

1. Launch SAP Web IDE. You can follow the steps in Configure and Check the SSO Configuration [page 1172].
2. Choose *File* > *New* > *Project from Template*, or on the SAP Web IDE welcome page, choose *New Project from Template* under *Create a Project*.
3. In the *Template Selection*, select *List Report Application*. Choose *Next*.
4. In the *Basic Information* section, add *TestExtension* as a project name and as a title. Choose *Next*.
5. In the *Data Connection* section select *Service Catalog*. Then, select a system from the drop-down menu. The destination you have created before should be there.
6. Select the destination you have defined. You should see a list of services.
7. If you don't get any errors, your destination with OAuth authentication is working, the test is fulfilled.
8. Exit the wizard.

5.2 Extending SAP SuccessFactors in the Neo Environment

You can extend the scope of SAP SuccessFactors HXM Suite using extension applications on SAP BTP.

This section guides you through the configuration tasks that you need to perform to enable the Neo environment for developing extension applications for your SAP SuccessFactors system.

> **i Note**
>
> The content in this section is only relevant for cloud management tools feature set A. For more information, see Cloud Management Tools - Feature Set Overview.

Overview

Extending SAP SuccessFactors on SAP BTP allows you to broaden the SAP SuccessFactors scope with applications running on the platform. This makes it quick and easy for companies to adapt and integrate SAP SuccessFactors cloud applications to their existing business processes, thus helping them maintain competitive advantage, engage their workforce and improve their bottom line.

> **i Note**
>
> You can integrate an extension subaccount that is part of a customer or partner global account in SAP BTP only. This functionality is not available for trial accounts.

Integration Layers

With the extension scenario you have the following integration layers between SAP BTP and SAP SuccessFactors:

- **Security and Connectivity**
  - SAML 2.0 single sign-on (SSO)
  - Integration between SAP SuccessFactors and the identity provider based on a trust configuration
  - Extension application connectivity interaction with SAP SuccessFactors based on the HXM Suite OData API
  - Extension application integration to consume events defined in the SAP SuccessFactors system
- **Development, Administration and Lifecycle Management**
  - Easy-to-configure connection between your account in SAP BTP and your SAP SuccessFactors system
  - Seamless solution deployment, subscription and monitoring
  - Development capabilities for building extensions that are tightly connected and integrated to SAP SuccessFactors by using all the authentication and authorization capabilities of the SAP BTP
- **User Interface Integration**
  With the integration between SAP BTP and SAP SuccessFactors you have a harmonized user experience across the standard SAP SuccessFactors modules and the new platform extensions.
  The SAP SuccessFactors Employee Central (EC) Home Page provides a framework that allows different modules to provide access to their functionality using tiles. For the extension applications hosted in the extension subaccount in SAP BTP, the cloud platform allows you to register Home Page tiles in the extended SAP SuccessFactors system.
  The integration also enables the SAP Cloud Portal service and configures it for extensibility so that you can create an extension site to integrate your extension application in SAP SuccessFactors.

Supported APIs

You can find a list and implementation details of the APIs supported by SAP SuccessFactors HXM Suite on SAP Help Portal, at SAP SuccessFactors HXM Suite OData API: Reference Guide.

Extension Application Installation

SAP BTP provides the following options for deploying and configuring SAP SuccessFactors extension application. The preferred option depends on your scenario.

- Deploying and configuring an extension application using Multi-Target Applications (preferable for productive scenarios).
  For more information, see .Multitarget Applications for the Neo Environment.
- Deploying and configuring an extension application using console client commands (preferable for development scenarios).
  For more information, see Installing and Configuring Extension Applications [page 1189].

Process Flow

You have to configure the cloud platform extension integration with SAP SuccessFactors to enable the use of applications running on top of the platform from SAP SuccessFactors.

These are the steps you need to follow:

1. Configuring the Subaccount in SAP BTP for SAP SuccessFactors [page 1177]
2. Installing and Configuring Extension Applications [page 1189]

5.2.1 Configuring the Subaccount in SAP BTP for SAP SuccessFactors

The following section contains detailed description of the steps you need to extend SAP SuccessFactors on SAP BTP. For more information about how to:

- Extend SAP SuccessFactors on SAP BTP, see Configure the Extension Integration Between SAP BTP and SAP SuccessFactors [page 1180].
- Enable the enhanced functionality of the Extension Management user interface (UI) available with the new SAP Cloud Portal service version, see Migrate to the New Version of SAP Cloud Portal Service [page 1184].
- Recover the extension artifacts and configuration settings after your cloud operators have performed an instance refresh, see:
  - Restore Configuration Settings After a Manual Instance Refresh [page 1187]
  - Restore Configuration Settings After an Automated Instance Refresh [page 1185]

> **i Note**
>
> You can integrate an extension subaccount that is part of a customer or partner global account in SAP BTP only. This functionality is available for enterprise accounts only.

5.2.1.1 Create an Integration Token for SAP SuccessFactors

You create an integration token required for the configuration of the extension integration between SAP BTP and SAP SuccessFactors.

Prerequisites

To create an integration token, you need to have assigned to your user either the Administrator predefined role or a custom platform role with the following scopes for the subbaccount which you want to integrate with your SAP BTP system:

- readExtensionIntegration
- manageExtensionIntegration
- manageDestinations
- manageApplicationRoleProvider
- manageTrustSettings

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

> **i Note**
>
> You can integrate an extension subaccount that is part of a customer or partner global account in SAP BTP only. This functionality is not available for trial accounts.

Context

To initiate the automated configuration for extending SAP SuccessFactors on SAP BTP, the SAP SuccessFactors administrators with Provisioning access need an integration token. It determines the subaccount in SAP BTP that will be integrated with your SAP SuccessFactors company, and with which your SAP SuccessFactors company will be linked.

> **i Note**
>
> One SAP SuccessFactors company can be integrated with more than one subaccount, while one subaccount can be associated with exactly one SAP SuccessFactors company.

As an SAP BTP user with permissions for the respective subaccount, you create an integration token using the SAP BTP cockpit, and then pass it over to the SAP SuccessFactors administrator.

Procedure

1. In your Web browser, open the cockpit using the relevant URL for the region with which your customer subaccount is associated. For more information about the regions, see Regions.
2. Select the relevant subaccount, and then choose *Integration Tokens* in the navigation area.

   > **i Note**
   >
   > This is the subaccount that is going to be associated with the SAP SuccessFactors system after a successful extension integration. You either use a new dedicated subaccount that you create in advance, or an already existing subaccount.

3. In the *Integration Tokens* panel, choose *New Token* > *New SAP SuccessFactors Token*.

   The *New SAP SuccessFactors Token* dialog box opens.
4. Select a trusted identity provider for single sign-on:
   - If you already have a configured trusted third-party identity provider, or an SAP Cloud Identity Services - Identity Authentication tenant in your extension subaccount that is marked as default one, you can keep it. To do that, select *Use existing custom identity provider*.
   - If you want to use the SAP SuccessFactors identity provider, select *Use SAP SuccessFactors identity provider*.

     > **i Note**
     >
     > If you select the *Use SAP SuccessFactors identity provider* option, after a successful integration, this identity provider will become the new default trust setting. This might affect Java/HTML5 applications that depend on the previous trust settings.

5. Choose *Create* and then *OK*.

   Your newly created token appears in the list of integration tokens and its status is `ACTIVE`. In the *Integration Tokens* panel, you can view details such as the user who has created the token, the creation date and the expiration date. The token is valid for 7 days after it has been created.

   > **i Note**
   >
   > The integration token can be used only once. Once the integration token is used, it is no longer valid.

6. (Optional) You can perform the following actions:
   - To view the identity provider used for the applications in the subaccount where the token has been created, choose *Token details* in the *Actions* column on the row of the respective token.
   - To delete an integration token, choose *Delete token* in the *Actions* column on the row of the respective token.

Results

You have created an integration token which you can use to initiate the automated configuration for extending SAP SuccessFactors on SAP BTP.

> **i Note**
>
> Make sure to use the integration token before its expiration date.

Next Steps

You can now pass over the value of the token to the SAP SuccessFactors administrator who will be triggering the automated configuration for extending SAP SuccessFactors on SAP BTP.

5.2.1.2 Configure the Extension Integration Between SAP BTP and SAP SuccessFactors

You configure the extension integration between SAP BTP and SAP SuccessFactors from Provisioning.

Prerequisites

- You have a dedicated SAP SuccessFactors company instance.
- You have Provisioning access.
- You have the integration token required to trigger the automated configuration. For more information, see Create an Integration Token for SAP SuccessFactors [page 1178].

  > **i Note**
  >
  > The integration token is used only to trigger the automated configuration and can be used only once. Once the integration token is used, it is no longer valid.

Context

Using Provisioning, you can configure the extension integration between your SAP SuccessFactors company and a subaccount in SAP BTP.

Procedure

1. Log on to Provisioning.
2. In the *List of Companies*, choose the company instance for which you want to configure the extension integration.

3. In the *Edit Company Settings* section, choose *Extension Management Configuration*.
4. In the *Integration Token* input field, enter the integration token you have received from your SAP BTP administrator.

   The integration token determines the subaccount that will be integrated with your SAP SuccessFactors company.

   > **i Note**
   >
   > One SAP SuccessFactors company can be integrated with more than one subaccount, while one subaccount can be associated with exactly one SAP SuccessFactors company.

5. Choose *Add*.
   In the *Extension Subaccounts Details* section, the system displays the progress of the automated configuration. To update the progress information, choose the *Check Status*. If the configuration of the extension package cannot complete in 120 minutes, the value in the *Integration Status* column is set to *ENABLE_FAILED*.

   By using the link in the *Extension Subaccount* column, you can view the configuration status of individual extension integration capabilities.

   In the *Extension Package Configuration Related Features Status* section, you can view the configuration status of individual extension package capabilities.

Results

After the automated configuration has finished:

- SAP Cloud Portal service is enabled (if it hasn't already been enabled) and configured for extensibility in your subaccount.
- SAP SuccessFactors is added as a default trusted application identity provider in your subaccount. A corresponding service provider is configured if there wasn't an already existing one.
- You have the `Extensions Administrators` group and the `Extensions Admin` permission role created in SAP SuccessFactors.

  > **⚠ Caution**
  >
  > You must not rename or delete the `Extensions Administrators` group and the `Extensions Admin` permission role. Otherwise, you will not be able to use the extension management UI in SAP SuccessFactors.

Next Steps

- Once you have performed the automated configuration, you include members in the *Extensions Administrators* group to define who can access the extension management UI from SAP SuccessFactors. For more information, see Defining the People Pool for Managing Extensions [page 1182].

- If you use the SAP SuccessFactors identity provider, you need to configure the access to SAP Web IDE Full-Stack. To do so, you assign developer permissions to the required users or groups. For more information, see Assign Users Permission for SAP Web IDE .
- Once you have performed the automated configuration of the extension integration, you continue with the installation and configuration of your extension applications. For more information, see Installing and Configuring Extension Applications [page 1189].

5.2.1.2.1 Defining the People Pool for Managing Extensions

Once you have performed the automated configuration, you include members in the *Extensions Administrators* group to define who can access the extension management page from SAP SuccessFactors.

Prerequisites

- You have the role-based permission environment enabled for the SAP SuccessFactors customer instance (company).
- You have a *Security Admin* user for SAP SuccessFactors and have access to the functionality on the SAP SuccessFactors *Admin Center* page.
- You have configured the SAP BTP extension integration with SAP SuccessFactors.

Context

The extension management page in SAP SuccessFactors is restricted to users who have the `Extensions Admin` role. This role together with the `Extensions Administrators` group to which it is granted, are created automatically during the cofiguration of the extension integration. To manage the users that have this role, see Using Role-Based Permissions.

> ⚠ Caution
>
> You must not rename or delete the `Extensions Administrators` group and the `Extensions Admin` permission role. Otherwise, you will not be able to use the extension management UI in SAP SuccessFactors.

5.2.1.3 Refresh the Extension Integration for SAP SuccessFactors

Prerequisites

You have Provisioning access.

Context

You can refresh the extension integration for the SAP SuccessFactors company and the respective subaccounts. To do so, in the Provisioning:

Procedure

1. Log on to Provisioning.
2. In the *List of Companies*, choose the company instance for which you want to configure the extension integration.
3. In the *Edit Company Settings* section, choose *Extension Management Configuration*.
4. In the *Extension Subaccounts Details* section, choose *Refresh* for the respective extension integration with a particular subaccount in SAP BTP.

Results

During the refresh:

- The trust between the subaccount in SAP BTP and the SAP SuccessFactors company instance is reconfigured, and generates a local service provider for this subaccount only if there isn't one.
- The *Authorized SP Assertion Consumer Service* settings are recreated for the extension integration.
- The OAuth clients for this extension integration are recreated.
- The OAuth SAML Bearer destinations created for the extension integration are updated.

> **i Note**
>
> If you have manually deleted an OAuth SAML Bearer destination with system user, after the refresh, this destination will be recreated with all the properties it used to have except the *System User*. For the *System User* property, it will be set the value `##REPLACE_USER##`. You need to manually edit this destination in the SAP BTP cockpit and replace the *System User* value with a real technical user.

- The *Extensions Admin* role and the *Extensions Administrators* group are recreated if they do not already exist, and the *Extensions Admin* role is reassigned to the *Extensions Administrators* group.

When the refresh is successful, the integration status of the extension subaccount is *ENABLED*.

5.2.1.4 Migrate to the New Version of SAP Cloud Portal Service

If your extension integration uses the old version of SAP Cloud Portal service, you need to migrate to the new version to enable an enhanced extension administration flow and user experience.

Context

In the new SAP Cloud Portal service, there are two available site templates for the SAP BTP extensions for SAP SuccessFactors.

As of September 2016, SAP SuccessFactors Q3 2016 supports the new version of SAP Cloud Portal service by default. If you want to use the enhanced SAP Cloud Portal service functionality with extension integration configured before then, you need to enable it manually.

Procedure

To enable the enhanced functionality of an extension integration that uses the old SAP Cloud Portal service, you refresh the respective extension integration with a particular subaccount in SAP BTP. For more information, see Refresh the Extension Integration for SAP SuccessFactors [page 1183].

Results

In the *Extensions* tool in *SAP SuccessFactors* > *Admin Center*, the Extension Directory link for the corresponding extension integration now points to the new SAP Cloud Portal service Admin Space.

You can now create extension sites using the enhanced functionality provided with the new SAP Cloud Portal service version. This version includes two site templates for the SAP SuccessFactors extensions for SAP SuccessFactors.

Related Information

Create an Extension Site [page 1201]

5.2.1.5 Restore Configuration Settings After an Automated Instance Refresh

If you have performed an automated instance refresh with the *Instance Refresh* tool, you restore some of your extension integration artifacts and configuration settings for the SAP SuccessFactors target company manually.

Prerequisites

- You have a dedicated SAP SuccessFactors company instance.
- You have Provisioning access and administrator permissions.
- You have integrated a subaccount in SAP BTP with your SAP SuccessFactors company.
- You have performed the instance refresh with the *Instance Refresh* tool.

Context

In SAP SuccessFactors, instance refresh is a procedure in which the data and the settings of a source company instance are copied into another company instance. During the instance refresh the data of the target company instance is deleted and replaced by the data of the source company instance. Therefore, after an instance refresh, if the target company has been integrated with a subaccount in SAP BTP, the extension integration configuration settings and artifacts in the target company are overwritten by the configuration settings and data coming from the source company.

The instance refresh can be triggered automatically with the *Instance Refresh* tool. If you have performed an instance refresh with the *Instance Refresh* tool, the OAuth clients, the Assertion Consumer Services (ACS) of the target company instance created by the cloud platform, and the OAuth SAML Bearer destinations, and the inbound connections are recreated for the extension integration.

> **i Note**
>
> The OAuth clients are recreated but they are assigned new IDs. This does not affect the configuration settings.

However, you must reconfigure the following artifacts and settings in the SAP SuccessFactors target company instance:

- Custom home page tiles
- Permission roles
- Permission groups
- Outbound connections (can be recreated with a solution deployment only)

The automated refresh results in the following behavior in the *Solutions* view in SAP BTP cockpit

- The custom home page tiles are missing.
- The OAuth clients cannot be found because they have new IDs after the refresh.

- The permission roles and permission groups are missing.
- The outbound connections are missing.

Procedure

1. To restore the configuration settings after an automated instance refresh, proceed as follows:
    - If you have used console client commands to configure the extension artifacts initially, proceed as follows:
        - To restore the home page tiles, register the home page tiles you want to restore. See Register a Home Page Tile for the Extension Application [page 1196]
        - To restore the permission roles and permission groups:
            1. Import the Extension Application Roles in the SAP SuccessFactors System [page 1205]
            2. Assign the Extension Application Roles to Users [page 1206]
            3. Test the Role Assignments [page 1209]
    - If you have used Multitarget Applications (MTA) to configure the settings and have a solution in which the home page tiles, the required permission roles and permission groups, and the outbound connections are defined, redeploy the solution. See Operating Solutions [page 1089].
2. Delete any unneeded extension integration configuration settings coming from the source company instance.

Results

- You have restored the extension integration artifacts and configuration setting of the target company instance and deleted any extension integration configuration settings coming from the source company instance.
- The *Solutions* view in SAP BTP cockpit displays the home page tiles.
- If you have redeployed a solution, the OAuth client IDs have been reverted to the initial ones and they can be found in the *Solution* view.

> **i Note**
>
> If you have used console client commands to restore the extension artifacts, the OAuth clients cannot be found in the *Solutions* view because they have new IDs after the refresh but they are restored.

5.2.1.6 Restore Configuration Settings After a Manual Instance Refresh

If your cloud operators have performed an instance refresh manually, you must restore your extension integration artifacts and configuration settings for the SAP SuccessFactors target company.

Prerequisites

- You have a dedicated SAP SuccessFactors company instance.
- You have Provisioning access and administrator permissions.
- You have integrated a subaccount in SAP BTP with your SAP SuccessFactors company.
- The cloud operators have performed an instance refresh manually.

Context

In SAP SuccessFactors, instance refresh is a procedure in which the data and the settings of a source company instance are copied into another company instance. During the instance refresh the data of the target company instance is deleted and replaced by the data of the source company instance. Therefore, after an instance refresh, if the target company has been integrated with a subaccount in SAP BTP, the extension integration configuration settings and artifacts in the target company are overwritten by the configuration settings and data coming from the source company. The instance refresh can be triggered either automatically with the *Instance Refresh* tool, or manually. For more information about restoring the extension artifacts after an automated instance refresh, see Restore Configuration Settings After an Automated Instance Refresh [page 1185].

The manual refresh is triggered by cloud operators. After your cloud operators have performed an instance refresh manually, you must trigger the operation for restoring the extension integration artifacts and configuration settings in the SAP SuccessFactors target company instance afterwards.

> **i Note**
>
> This operation includes restoring the OAuth clients, the Assertion Consumer Services (ACS) of the target company instance created by the cloud platform, the *Extensions Admin* role and the *Extensions Administrators* group, and the outbound connections. The OAuth SAML Bearer destinations are recreated for the extension integration.

To restore the extension artifacts after a manual instance refresh, proceed as follows:

Procedure

1. Log on to Provisioning.
2. In the *List of Companies*, choose the company instance for which you want to recover the extension integration artifacts and configuration settings.

3. In the *Edit Company Settings* section, choose *Extension Management Configuration*.
4. Choose *Prepare Restore*.

> **i Note**
>
> This section appears only after an instance refresh has been performed.

5. Choose *Restore* for the respective extension subaccounts to restore their integration.

Results

You have restored the OAuth clients, the Assertion Consumer Services (ACS) of the target company instance created by the cloud platform, the *Extensions Admin* role and the *Extensions Administrators* group, and the outbound connections for the target company instance and deleted any extension integration configuration settings coming from the source company instance.

The following artifacts need to be restored manually:

- Home page tiles
- All permission roles but the default *Extensions Admin*.
- All permission groups but the default *Extensions Administrators*.

In the *Solutions* view in SAP BTP cockpit

- The home page tiles are missing.
- The OAuth clients cannot be found because they have new IDs after the refresh.
- The permission roles and permission groups are missing.

Next Steps

- If you have used console client commands to configure the extension artifacts initially, proceed as follows to restore the home page tile, and the permission roles and groups:
  - To restore the home page tiles, register the home page tiles you want to restore. See Register a Home Page Tile for the Extension Application [page 1196]
  - To restore the permission roles and permission groups:
    1. Import the Extension Application Roles in the SAP SuccessFactors System [page 1205]
    2. Assign the Extension Application Roles to Users [page 1206]
    3. Test the Role Assignments [page 1209]
- If you have used Multitarget Applications (MTA) to configure the settings and have a solution in which the home page tiles, and the required permission roles and permission groups are defined, redeploy the solution. See Operating Solutions [page 1089].

5.2.1.7 Remove the Extension Integration

Context

You can remove the extension integration between your SAP SuccessFactors company and the subaccount in SAP BTP. This will remove:

- the company identity provider from the trusted identity providers in the subaccount. Only the identity providers that have been added during the configuration of the extension integration will be removed.
- the company OAuth clients managed by the SAP BTP
- all company Authorized SP Assertion Consumer Services (ACS) managed by SAP BTP

> **i Note**
>
> This will affect all applications that use the extension integration.

Procedure

1. Log on to Provisioning.
2. In the *List of Companies*, choose the company instance for which you want to remove the extension integration.
3. In the *Edit Company Settings* section, choose *Extension Management Configuration*.
4. Choose *Remove*.
   In the *Extension Subaccounts Details* section, the system displays the progress of the remove operation. To update the progress information, choose *Check Status*. When the extension integration is removed, the entry disappears from the list.

5.2.2 Installing and Configuring Extension Applications

As an implementation partner, you install and configure the extension applications that you want to make available for customers.

You deploy your extension application, configure its connectivity to the SAP SuccessFactors system and map the roles defined in your extension application to the roles in the corresponding SAP SuccessFactors system.

Prerequisites

- You have an SAP BTP extension subaccount and the corresponding SAP SuccessFactors customer instance connected to it. For more information about extension subaccounts, see Extensions.

- You are either an administrator of the SAP BTP subaccount or have a platform role with the following platform scopes defined for it:
  - manageExtensionConfigurations
  - readExtensionConfigurations
  - readJavaApplications
  - readSubscriptions
  - readAccount
  - manageDestintaions
  - readDestinations
  - readJavaApplications
  - readHTML5Applications
  - manageApplicationRoleProvider
  - readSubscriptionConfiguration

Process Flow

You deploy your extension application in your SAP BTP extension. Then you need to configure the application connectivity to SAP SuccessFactors and to enable the use of the HXM Suite OData API. To ensure that only approved applications are using the SAP SuccessFactors identity provider for authentication, you need to register the extension application as an authorized assertion consumer service in SAP SuccessFactors. Then you register the extension application home page tiles and import the extension application roles in the SAP SuccessFactors customer instance connected to the extension subaccount.

To finalize the configuration on SAP BTP side, you change the default role provider to the SAP SuccessFactors one. To finalize the configuration on SAP SuccessFactors side, you assign user groups to the permission roles defined for your extension application.

> **i Note**
>
> In the Neo environment, you can alternatively deploy your extension application as a solution. You can package all artifacts that are needed for your extension solution – Java applications, HTML5 application, database bindings, destinations, even SAP SuccessFactors roles and home page tiles into a Multi-Target Application (MTA) archive that includes a deployment descriptor file, and then deploy the MTA archive in your account. See:
>
> - Multitarget Applications for the Neo Environment
> - Modeling SAP SuccessFactors Extensions

Task	Description
1. Register the Extension Application as an Authorized Assertion Consumer Service [page 1191]	Register the extension application as an authorized assertion consumer service.
2. Configure the Extension Application's Connectivity to SAP SuccessFactors [page 1194]	Configure the connectivity between your Java extension application and the SAP SuccessFactors system associated with your SAP BTP extension subaccount.

Task	Description
3. Register a Home Page Tile for the Extension Application [page 1196]	Register a home page tile for the extension application in the extended SAP SuccessFactors system
4. Create the Resource File with Role Definitions [page 1202]	Create the resource file containing the SAP SuccessFactors HXM role definitions.
5. Import the Extension Application Roles in the SAP SuccessFactors System [page 1205]	Import the application-specific roles from the SAP BTP system repository into to the SAP SuccessFactors customer instance connected to your extension subaccount.
6. Assign the Extension Application Roles to Users [page 1206]	Assign the extension application roles you have imported in the SAP SuccessFactors systems to the user to whom you want to grant access to your application.
7. Configure the SAP SuccessFactors Role Provider [page 1207]	Change the default SAP BTP role provider of your Java application to the SAP SuccessFactors role provider. **i Note** This task is relevant for Java extension applications only.
8. Test the Role Assignments [page 1209]	Try to access the application with the users with different level of granted access to test the role assignments.

5.2.2.1 Register the Extension Application as an Authorized Assertion Consumer Service

Register the extension application as an authorized assertion consumer service to configure its access to the SAP SuccessFactors system through the SAP SuccessFactors identity provider (IdP).

Prerequisites

- The SAP BTP subaccount in which you configure the connectivity to the SAP SuccessFactors system is an extension subaccount. For more information about extension subaccounts, see Extensions.
- You are either an administrator of the subaccount or have the developer platform role with the following platform scopes defined for it:
  - readExtensionIntegration
  - manageExtensionIntegration
  - readHTML5Applications

Context

Extension applications deployed in an SAP BTP extension subaccount are authenticated against the SAP SuccessFactors (IdP). To ensure that only approved applications are using the SAP SuccessFactors IdP for

authentication, you need to register the extension application as an authorized assertion consumer service, configure the application URL, the service provider audience URL and the service provider logout URL of the extension application in SAP SuccessFactors Provisioning. To do so you use the `hcmcloud-enable-application-access` console client command.

> **i** Note
>
> This procedure is only valid for connections using the default SAP SuccessFactors.

Procedure

1. Open the command prompt and navigate to the folder containing neo.bat/sh (SDK installation folder/tools).
2. Register the extension application as an authorized assertion consumer service. In the console client command line, execute: **`hcmcloud-enable-application-access`**, as follows:
   - For an application deployed in your subaccount, specify the name of your extension application for the `application` parameter.
     For example, to register a Java extension application running in your subaccount in the US East region, execute:

     ```
     neo hcmcloud-enable-application-access --application <my_application> --
     application-type java --account <my_extension_subaccount> --user
     <my_email@example.com> --host us1.hana.ondemand.com
     ```

   - For an application to which your subaccount is subscribed, specify the application provider subaccount and the name of your extension application for the `application` parameter in the following format: `<application_provider_subaccount>:<my_application>`.
     For example, to register a Java extension application to which your subaccount in the US East region is subscribed, execute:

     ```
     neo hcmcloud-enable-application-access --application
     <application_provider_subaccount:my_application> --application-type java --
     account <my_extension_subaccount> --user <my_email@example.com> --host
     us1.hana.ondemand.com
     ```

3. (Optional) Display the status of an application entry in the list of authorized assertion consumer services for the SAP SuccessFactors system associated with an extension subaccount. In the console client command line, execute **`hcmcloud-display-application-access`**, as follows:
   - For an application deployed in your subaccount, specify the name of your extension application for the `application` parameter.
     For example, to display the status of the authorized assertion consumer service entry for an application deployed in your subaccount in the US East region, execute:

     ```
     neo hcmcloud-display-application-access-status --application
     <my_application> --application-type java --account
     <my_extension_subaccount> --user <my_email@example.com> --host
     us1.hana.ondemand.com
     ```

   - For an application to which your subaccount is subscribed, specify the application provider subaccount and the name of your extension application for the `application` parameter in the following format: `<application_provider_subaccount>:<my_application>`.

For example, to display the status of the authorized assertion consumer service entry for an application to which your subaccount in the US East region is subscribed, execute:

```
neo hcmcloud-display-application-access-status --application
<application_provider_subaccount>:<my_application> --application-type java
--account <my_extension_subaccount> --user <my_email@example.com> --host
us1.hana.ondemand.com
```

4. (Optional) If your scenario requires it, remove the entry of the extension application from the list of authorized assertion consumer services for the SAP SuccessFactors system associated with the extension subaccount. In the console client command line, execute **`hcmcloud-disable-application-access`**, as follows:

   - For an application deployed in your subaccount, specify the name of your extension application for the `application` parameter.
     For example, to remove the authorized assertion consumer service entry for a Java application deployed in your subaccount in the US East region, execute:

     ```
     neo hcmcloud-disable-application-access --application <my_application> --
     application-type java --account <my_extension_subaccount> --user
     <my_email@example.com> --host us1.hana.ondemand.com
     ```

   - For an application to which your subaccount is subscribed, specify the application provider subaccount and the name of your extension application for the `application` parameter in the following format: `<application_provider_subaccount>:<my_application>`.
     For example, to remove the authorized assertion consumer service entry for a Java application to which your subaccount in the US East region is subscribed, execute:

     ```
     neo  hcmcloud-disable-application-access --application
     <application_provider_subaccount>:<my_application> --application-type java
     --account <my_extension_subaccount> --user <my_email@example.com> --host
     us1.hana.ondemand.com
     ```

Related Information

hcmcloud-enable-application-access [page 1461]
hcmcloud-disable-application-access [page 1458]
hcmcloud-display-application-access-status [page 1459]

5.2.2.2 Configure the Extension Application's Connectivity to SAP SuccessFactors

Use this procedure to configure the connectivity between your Java or HTML5 extension application and the SAP SuccessFactors system associated with your SAP BTP extension subaccount.

Prerequisites

- You have an SAP BTP extension subaccount and the corresponding SAP SuccessFactors customer instance connected to it.
- You are either an administrator of the subaccount or have a platform role with the following platform scopes defined for it:
  - readExtensionIntegration
  - manageExtensionIntegration
  - manageDestinations

Context

The extension applications interact with the extended SAP SuccessFactors system using the HXM Suite OData API. The HXM Suite OData API is a RESTful API based on the OData protocol intended to enable access to data in the SAP SuccessFactors system. You can benefit from the following the following API access scenarios:

- OData access with SAML2BearerAssertion authentication
  This scenario is used for performing OData API calls with logged-in user propagation, thus enforcing permission checks for accessing objects. For this scenario, you require a `sap_hcmcloud_core_odata` destination.
- OData access with SAML2BearerAssertion authentication and a technical user
  This scenario is used for performing OData API calls with a predefined technical user when the extension application is performing scheduled jobs or data replication. For this scenario you require a `sap_hcmcloud_core_odata_technical_user` destination.

To enable the API access and configure the connectivity between the Java or HTML5 extension applications and the SAP SuccessFactors system associated with your extension subaccount, you use the `hcmcloud-create-connection` console client command. Using the command, you specify the connection details for the remote communication of the extension application and create the HTTP destinations required for configuring the API access. The command also creates and configures the corresponding OAuth clients in the SAP SuccessFactors company instance.

You can consume this destination in your application using one of these APIs:

- HttpDestination API and DestinationFactory
- ConnectivityConfiguration API
- AuthenticationHeaderProvider API

> **Note**
>
> As of August 2019, we have introduced the `skipUserAttributesResolution` destination property which improves the performance. If you want to enable the property for connections created before the property has been introduced, you can do one of the following:
>
> - Refresh the extension integration. For more information, see Refresh the Extension Integration for SAP SuccessFactors [page 1183].
> - For the connection for which you want to enable the `skipUserAttributesResolution` property, execute the **`hcmcloud-create-connection`** command with the `--overwrite` parameter specified. For more information, see: hcmcloud-create-connection.
> - Edit the destination and add the `skipUserAttributesResolution` property manually and set its value to `true`. For more information, see: Managing Destinations.

You have the following options for configuring the connectivity:

- (Relevant for Java applications only) If you have an extension application deployed in your subaccount, you can configure the connectivity on an application level in the subaccount where the application is deployed.
- (Relevant for Java applications only) If your subaccount is subscribed to an extension application, you can configure the connectivity on a subscription level in the subaccount subscribed to the application.
- If you want to create a connection that can be used by all the extension applications that are deployed in your subaccount or that your subaccount is subscribed to, you configure the connectivity on a subaccount level.

  > **Note**
  >
  > If you have an HTML5 extension application, use this option.

You can optionally list the connections created for the extension application. You can also delete a connection.

Procedure

1. To configure the connectivity, in the console client command line, execute the **`hcmcloud-create-connection`** command, as described in hcmcloud-create-connection [page 1454].
2. (Optional) To list the connections created for the extension application, in the console client command line, execute **`hcmcloud-list-connections`**, as described in hcmcloud-list-connections [page 1468].
   - For an application deployed in your subaccount, specify the name of your extension application for the `application` parameter.
   - For an application to which your subaccount is subscribed, specify the application provider subaccount and the name of your extension application for the `application` parameter in the following format: `<app_provider_subaccount>:<my_app>`.
   - For a subaccount level connectivity, do not use the `application` parameter.
3. (Optional) If your scenario requires it, remove the connectivity configured between your extension application and the SAP SuccessFactors systems associated with the extension subaccount. In the console client command line, execute **`hcmcloud-delete-connection`**, as described in hcmcloud-delete-connection [page 1456].

- For an application deployed in your subaccount, specify the name of your extension application for the `application` parameter.
- For an application to which your subaccount is subscribed, specify the application provider subaccount and the name of your extension application for the `application` parameter in the following format: `<app_provider_subaccount>:<my_app>`.
- For a subaccount level connectivity, do not use the `application` parameter.

Related Information

hcmcloud-delete-connection [page 1456]
hcmcloud-create-connection [page 1454]
hcmcloud-list-connections [page 1468]

5.2.2.3 Register a Home Page Tile for the Extension Application

You register a Home Page tile for the extension application in the extended SAP SuccessFactors system so that you can access the application directly from the SAP SuccessFactors Employee Central (EC) Home Page.

Prerequisites

- You have deployed and started the extension application for which you are registering the Home Page tile.
- You have registered the extension application as an authorized assertion consumer service. For more information, see Register the Extension Application as an Authorized Assertion Consumer Service.
- You have the Home Page tile provided as part of the application interface.
  You develop the content of the tile as a dedicated HTML page inside the application and size it according to the desired tile size. You describe the tiles in a `tiles.json` descriptor and package them in a `ZIP` archive. For more information about the structure of the `tiles.json` descriptor, see Home Page Tiles JSON File [page 1198].
- You have created the `tiles.json` descriptor.

Context

The SAP SuccessFactors EC Home Page provides a framework that allows different modules to provide access to their functionality using tiles. For the extension applications hosted in the SAP BTP extension subaccount, SAP BTP allows you to register Home Page tiles in the extended SAP SuccessFactors system. To do so, you use the `hcmcloud-register-home-page-tiles` console client command. Both Java and HTML5 extension applications are supported.

Procedure

1. Open the command prompt and navigate to the folder containing `neo.bat/sh` (SDK installation folder/tools).
2. Register the SAP SuccessFactors EC Home Page tiles in the SAP SuccessFactors company instance linked to the specified SAP BTP subaccount. In the console client command line, execute: **`hcmcloud-register-home-page-tiles`**, as follows:
   - For an application deployed in your subaccount, specify the name of your extension application for the `application` parameter.
     For example, to register a Home Page tile for a Java extension application running in your subaccount in the US East region, execute:

     ```
     neo hcmcloud-register-home-page-tiles --application <my_application> --
     application-type java --account <my_extension_subaccount> --user
     <my_email@example.com> --host us1.hana.ondemand.com --location <path to
     the tiles.json file>
     ```

   - For an application to which your subaccount is subscribed, specify the application provider subaccount and the name of the extension application for the `application` parameter in the following format: `<application_provider_subaccount>:<my_application>`.
     For example, to register a Home Page tile for a Java extension application to which your subaccount in the US East region is subscribed, execute:

     ```
     neo hcmcloud-register-home-page-tiles --application
     <application_provider_subaccount:my_application> --application-type java --
     account <my_extension_subaccount> --user <my_email@example.com> --host
     us1.hana.ondemand.com --location <path to the tiles.json file>
     ```

   > **i Note**
   >
   > The size of the tile descriptor file must not exceed 100 KB.

3. (Optional) List the extension application Home Page tiles registered in the SAP SuccessFactors company instance associated with the extension subaccount. In the console client command line, execute **`hcmcloud-get-registered-home-page-tiles`**, as follows:
   - For an application deployed in your subaccount, specify the name of your extension application for the `application` parameter.
     For example, to list the tiles registered for a Java extension application deployed in your subaccount in the US East region, execute:

     ```
     neo hcmcloud-get-registered-home-page-tiles --application <my_application>
     --application-type java --account <my_extension_subaccount> --user
     <my_email@example.com> --host us1.hana.ondemand.com
     ```

   - For an application to which your subaccount is subscribed, specify the application provider subaccount and the name of the extension application for the `application` parameter in the following format: `<application_provider_subaccount>:<my_application>`.
     For example, to list the tiles registered for a Java extension application to which your subaccount in the US East region is subscribed, execute:

     ```
     neo hcmcloud-get-registered-home-page-tiles --application
     <application_provider_subaccount>:<my_application> --application-type java
     ```

```
--account <my_extension_subaccount> --user <my_email@example.com> --host
us1.hana.ondemand.com
```

> **i Note**
>
> If you do not specify the `application` parameter, the command returns all the tiles registered in the SAP SuccessFactors EC Home Page of the SAP SuccessFactors company instance linked to the extension subaccount.
>
> There is no lifecycle dependency between the tiles and the application, so the application may not be started or may not be deployed anymore.

4. (Optional) If your scenario requires it, unregister the SAP SuccessFactors EC Home Page tiles registered for the extension application. In the console client command line, execute **`hcmcloud-unregister-home-page-tiles`**, as follows:

   - For an application deployed in your subaccount, specify the name of your extension application for the `application` parameter.
     For example, to unregister the SAP SuccessFactors EC Home Page tiles for a Java application deployed in your subaccount in the US East region, execute:

     ```
     neo hcmcloud-unregister-home-page-tiles --application <my_application> --
     application-type java --account <my_extension_subaccount> --user
     <my_email@example.com> --host us1.hana.ondemand.com
     ```

   - For an application to which your subaccount is subscribed, specify the application provider subaccount and the name of your extension application for the `application` parameter in the following format: `<application_provider_subaccount>:<my_application>`.
     For example, to unregister the SAP SuccessFactors EC Home Page tiles for a Java application to which your subaccount in the US East region is subscribed, execute:

     ```
     neo hcmcloud-unregister-home-page-tiles --application
     <application_provider_subaccount>:<my_application> --application-type java
     --account <my_extension_subaccount> --user <my_email@example.com> --host
     us1.hana.ondemand.com
     ```

> **i Note**
>
> There is no lifecycle dependency between the tiles and the application, so the application may not be started or may not be deployed anymore.

5.2.2.3.1 Home Page Tiles JSON File

The Home Page tiles JSON descriptor, for example, `tiles.json`, contains the definition of the Home Page tiles for the extension application.

Properties

Home Page Tile Properties	
`name`	The name of the tile used to identify it. This name is used in the Home Page administration tools and it is not visible to end-users, but only to HR administrators.
`metadata`	Defines the localized tile title and description.
• `title`	A metadata property. This is the title of the custom tile as it appears to end-users on the Home Page. The value *en_US* is mandatory. Otherwise, the value for other locales is not provided, and end-users will see a blank tile.
• `subTitle`	A metadata property. This is a subtitle that appears on custom tiles under the tile title. Subtitles are optional. If you do not want to use a subtitle, you can leave the field blank.
• `locale`	A metadata property. Using this propety, you can localize the title, and the subtitle.
`type`	Determines how the tile appears to end-users. Currently, only static type is supported. The value *static* is mandatory.
`configuration`	Contains the *icon* property.
• `icon`	A configuration property. Contains the ID of the icon that you want to use for the custom tile. You can take its ID from SAP SuccessFactors system. Go to *Admin Center* > *Manage Home Page* > *Add Custom Tile* and then follow the wizard until you choose the icon in the *Tile* tab. Then, take its ID and place it in the `tiles.json`. For example, "sap-icon://add-product".

Home Page Tile Properties	
`navigation`	The tile navigation determines how the tile responds when a user clicks or selects it. You can choose from the following options: • type: html-popover configuration: Contains two properties, "contentSize" and "content". "contentSize" defines the width of the popover window and has one of the following values: "responsive", "small", "medium", and "large". "content" defines the HTML content of the popover window and has a String value. You cannot localize the content of this type (unlike the SAP SuccessFactors custom popover tiles). • type: iframe-popover configuration: Contains two properties, "contentSize" and "URL". "contentSize" defines the size of the popover window and has one of the following values: "responsive", "small", "medium", and "large". "URL" defines the relative to the application root of the iframe source and has a String value. • type: url configuration: Contains the "url" property. "url" defines the URL link which will be opened and has a String value. For example, "index.html". The "url" is relative to the application root. • type: no_target

For more details about the Home Page tiles properties, see Custom Tile Configuration Settings.

> **i Note**
>
> The `tiles.json` descriptor file must use UTF-8 encoding and its size must not exceed 100 KB.

Example

Example of a Home Page tile JSON descriptor.

```
[{
    "tileVersion": "NEW_HOME_PAGE",
    "tiles": [{
            "name": "New Test Application",
            "section": "news",
            "metadata": [{
                "title": "My new home page tile",
                "subTitle": "This is new home page tile",
                "locale": "en_US"
            }],
            "type": "static",
            "configuration": {
                "icon": "sap-icon://add-product"
            },
            "navigation": {
                "type": "url",
                "configuration": {
```

```
                    "openInNewWindow": false,
                    "url": "index.html"
                }
            }
        }]
}]
```

5.2.2.4 Create an Extension Site

You create an extension site to integrate the extension application in SAP SuccessFactors.

Prerequisites

- You have administrator's permissions for SAP SuccessFactors. See Defining the People Pool for Managing Extensions [page 1182].
- For Java extension applications, make sure that the application's client-side is developed as a SAPUI5 component.
  HTML5 extension applications are supported by default.

Context

Procedure

1. In *SAP SuccessFactors* > *Admin Center*, navigate to *Company Settings* > *Extensions*.
2. In *Extension Directory*, click the + tile to add a new extension site.
3. In the extension site creation wizard:
   a. Enter a name.
   b. In the *Template Source* dropdown list, choose *SAP SuccessFactors Extensions*.
   c. Select one of the templates based on your preferred layout.
   d. Choose *Create*.

   Your new extension site opens in edit mode.
4. 4. Add your extension application to the extension site. This extension application has to be an HTML5 application.
   1. From the *Site Designer* side panel, open *Content Management*, and choose *Apps*.
   2. Choose + at the bottom left side, and choose *New*.
   3. From the *App Resource* field, search for your extension application, and choose *OK*.

   The application is added to the *Apps* list.

5. 5. Add the extension application to the site menu.
    1. In the *Site Designer* side navigation panel, choose *Menu Editor*.
    2. Choose + and select *Add Item*.
    3. In the *Add Menu Item* dialog, select *App* as your *Item Type*.
    4. Search for your application and add it as a full-page application in your site.
    5. Choose *Save*.

    If you preview your extension site now, you'll see that the application is part of the site menu and is displayed as a full page application in your site.
6. Publish the extension site to make it available to all users.
    a. Choose *Extension Directory*.
    b. Hover over the card of your site, and click the arrow at the bottom right corner.
    c. Choose *Publish*.
7. Access the extension site at runtime.
    a. Hover over the published card of your site.
    b. Choose the runtime URL that is displayed on the card.

Results

You have created an extension site for your extension application, and your extension application appears in it.

> **i** Note
>
> The SAP Cloud Portal service documentation uses the generic term *Site* and not *Extension Site*. Both are referring to the same entity.

5.2.2.5 Create the Resource File with Role Definitions

You create the resource file containing the SAP SuccessFactors HXM role definitions.

Prerequisites

- The corresponding SAP SuccessFactors HXM Suite roles exist in the SAP SuccessFactors system.
- You have admin access to the SAP SuccessFactors OData API and have a valid subaccount with user name and password.

Context

To create the resource file with the role definitions required for your application, you use the SAP SuccessFactors OData API to query the permissions defined for this role, and create a `roles.json` file containing the role definitions. You use HTTP Basic Authentication for the OData API call.

Procedure

1. Call the OData API to query the permissions defined for the required role using the following URL:

   `https://<SAP_SuccessFactors_host_name>/odata/v2/RBPRole?$filter=roleName eq '<role_name>'&$expand=permissions&$format=json`

   Where:

   - `<host_name>` is the fully qualified domain name of the OData API host depending on the region hosting your SAP SuccessFactors instance. For more information about the OData API endpoints, see https://help.sap.com/viewer/d599f15995d348a1b45ba5603e2aba9b/LATEST/en-US/03e1fc3791684367a6a76a614a2916de.html.
   - `<role_name>` is the name of the role as defined in the SAP SuccessFactors system.

   The response is a JSON object containing the following properties for each of the permissions defined for the specified role:

Property	Include in `roles.json`	Description
permissionID	No	(Key)Permission ID, the sequence ID in permission table
permissionType	Yes	Permission_Type column
permissionStringValue	Yes	Permission_string_value column
permissionLongValue	Yes	Permission_long_value column

Example response

```
{
    "d": {
        "__metadata": {
            "uri": "https://localhost:443/odata/v2/RBPRole(82L)",
            "type": "SFOData.RBPRole"
        },
        "roleId": "82",
        "roleDesc": "Testing role permissions",
        "lastModifiedBy": "admin",
        "lastModifiedDate": "\/Date(1404299328000)\/",
        "roleName": "Test Role Permissions",
        "userType": "null",
        "permissions": {
            "results": [{
                "__metadata": {
                    "uri": "https://localhost:443/odata/v2/
RBPBasicPermission(60L)",
                    "type": "SFOData.RBPBasicPermission"
                },
                "permissionId": "60",
```

```
                "permissionType": "user_admin",
                "permissionStringValue": "change_info_user_admin",
                "permissionLongValue": "-1"

            },
            {
                "__metadata": {
                    "uri": "https://localhost:443/odata/v2/
RBPBasicPermission(4L)",
                    "type": "SFOData.RBPBasicPermission"
                },
                "permissionId": "4",
                "permissionStringValue": "detail_report",
                "permissionLongValue": "-1",
                "permissionType": "report"
                }]
]
        }
    }
}
```

2. Create a `roles.json` file using the following properties. To list all the available permissions in your SAP SuccessFactors system, use this OData API call: **`https://<SAP_SuccessFactors_host_name>/odata/v2/RBPBasicPermission?$format=json`**. There you can find the properties that you need to create the `roles.json` file.

Property	Description
`roleName`	Name of the role as defined in the response to the OData API call
`roleDesc`	Role description as defined in the response to the OData API call
`permissionType`	Value of the `permissionType` property as defined in the response to the OData API call
`permissionStringValue`	Value of the `permissionStringValue` property as defined in the response to the OData API call
`permissionLongValue`	Value of the `permissionLongValue` property as defined in the response to the OData API call

Target `roles.json` file

```
[{
            "roleDesc": "My role description",
            "roleName": "My Application Role Name",
            "permissions": [{
                "permissionStringValue": "change_info_user_admin",
                "permissionLongValue": "-1",
                "permissionType": "user_admin"
            },
            {
                "permissionStringValue": "detail_report",
                "permissionLongValue": "-1",
                "permissionType": "report"
    }]
}]
```

3. Save the file locally.

Results

You have created the role definition resource file.

5.2.2.6 Import the Extension Application Roles in the SAP SuccessFactors System

To complete the authorization configuration of your extension application, you import the application-specific roles into to the SAP SuccessFactors company instance connected to your extension subaccount.

Prerequisites

- You have created the resource file with the required role definitions. For more information, see Create the Resource File with Role Definitions [page 1202].
- You have downloaded and configured SAP BTP console client. For more information, see Using the Console Client [page 1348].

Context

Using the `hcmcloud-import-roles` console client command, you import the required role definitions in the SAP SuccessFactors company instance connected to this account.

Procedure

1. Open the command prompt and navigate to the folder containing neo.bat/sh (SDK installation folder/tools).
2. Execute the following command:

```
neo hcmcloud-import-roles --account <subaccount_name> --user <e-mail_or_user>
--host <host> --location <path to the file containing role definitions>
```

> **i Note**
>
> For more information about the file that contains the role definitions, see Create the Resource File with Role Definitions [page 1202].

> **i Note**
>
> The size of the file containing the role definitions must not exceed 500 KB.

3. Enter your password if requested.
4. Press ENTER and the import of the role definitions starts.

Results

You have imported the application-specific roles in the SAP SuccessFactors company instance connected to your subaccount. Now you need to assign users to these roles.

Related Information

Assign the Extension Application Roles to Users [page 1206]
hcmcloud-import-roles [page 1466]

5.2.2.7 Assign the Extension Application Roles to Users

To complete the authorization configuration for your extension application, you assign the extension application roles you have imported in the SAP SuccessFactors systems to the user to whom you want to grant access to your application.

Prerequisites

- You have a role-based permission environment for your SAP SuccessFactors company instance.
- Your have either a *Super Administrator* or a *Security Admin* user for SAP SuccessFactors and have access to the functionality on the SAP SuccessFactors Admin page.
- You have deployed the extension application.

Context

Use this procedure to assign the permission roles to users.

Procedure

1. Log on to SAP SuccessFactors with the following URL:

   ```
   https://<SAP_SuccessFactors_landscape>/login
   ```

Where `<SAP_SuccessFactors_landscape>` is the fully qualified domain name of the host on which the SAP SuccessFactors company is running.

2. Navigate to the *Manage Permission Roles*, as follows:
   - For Version 12 UI Framework (Revolution) not enabled: Navigate to: *Admin Center* > *Manage Security* > *Manage Permission Roles*.
   - For Version 12 UI Framework (Revolution) enabled: Navigate to: *Admin Center* > *Manage Employees* > *Set User Permissions* > *Manage Permission Roles*.
3. Locate the role you want to manage, and from the *Take Action* dropdown box next to the role, select *Edit*.
4. On the *Permission Role Detail* page, scroll down to the *Grant this role to...*section, and then choose *Add*. The system opens the *Grant this role to...* page.
5. On the *Grant this role to...* page, define whom you want to grant this role to, and specify the target population accordingly.
6. To navigate back to the *Permission Role Detail* page, choose *Finished*.
7. Save your entries.

5.2.2.8 Configure the SAP SuccessFactors Role Provider

If you have a Java extension application, you can change the default SAP BTP role provider to the SAP SuccessFactors role provider.

Prerequisites

- You have an SAP BTP extension subaccount and the corresponding SAP SuccessFactors customer instance connected to it. For more information about extension subaccounts, see Extensions.
- You are either an administrator of the subaccount or have a platform role with the following platform scopes defined for it:
- You have a Java extension application.

Context

A role provider is the component that retrieves the roles for a particular user. By default, the role provider used for SAP BTP applications and services is the SAP BTP role provider. For Java extension applications, however, you can change the default role provider to the SAP SuccessFactors role provider. Depending on whether the application is running in your subaccount or your subaccount is subscribed to the extension application, you configure the role provider from either the *Roles* section for your application, or the *Subscription* section for your subaccount. In addition, you can view the role provider for each enabled SAP BTP service in the *Services* section of the SAP BTP cockpit.

Alternatively, you can use the `hcmcloud-enable-role-provider` console client command. For more information, see hcmcloud-enable-role-provider [page 1463].

> **Note**
>
> Currently, the automated change of the role provider is available only for Java extension applications for SAP SuccessFactors.

Procedure

1. In the SAP BTP cockpit, navigate to the application for which you want to change the role provider. To do so, proceed as follows:
   - For a Java application running in your subaccount, choose *Applications* > *Java Applications*, and then choose the link of the application.
   - For a Java application to which your subaccount is subscribed, choose *Applications* > *Subscriptions*, and then choose the link of the application.
2. Choose *Security* > *Roles*.
3. In the *Role Provider* panel, select the required role provider from the *Provider* dropdown box.
4. (Optional) To view the role provider for an SAP BTP service, in the cockpit navigate to *Services* > `<service_name>`, and then choose *Configure Roles*.

   The system displays the role provider in the *Role Provider* panel in a read-only mode.

   > **Note**
   >
   > For an extension subaccount, the role provider for SAP Cloud Portal service is *SAP SuccessFactors*.

Results

The changes take effect after 5 minutes. If you want the changes to take effect immediately, you restart the application (valid only for applications running in your subaccount).

Related Information

hcmcloud-enable-role-provider [page 1463]

5.2.2.9 Test the Role Assignments

To test the role assignments you first start the deployed extension application to make it available for requests, and then try to access it with the users with different level of granted access to the application.

Prerequisites

- You have downloaded and configured SAP BTP console client. For more information, see Console Client for the Neo Environment [page 1348].
- You have made yourself familiar with the SAP BTP cockpit concepts. For more information, see Account Administration in the Cockpit [page 1259].

Context

Procedure

1. Open the command prompt and navigate to the folder containing `neo.bat/sh` (`<SDK installation folder>/tools`).
2. Start the deployed application using the following command:

   ```
   neo start --account <subaccount_name> --application <application_name> --user <e-mail_or_user> --host <host>
   ```

3. Access the application using users with different roles assigned to them.

   To access the application, use the application URL. To get the login URL of an application deployed in your extension subaccount, open the SAP BTP cockpit, and navigate to *Account* > `<subaccount_name>` > *Java Applications* > `<name_of_your_extension_application>` > *Application URLs*.

5.2.2.10 Integrate the Extension Application with SAP SuccessFactors Intelligent Services

You can integrate the extension application to subscribe to an event defined in the SAP SuccessFactors system.

With SAP BTP you can choose between the following integration flows that enable your extension application to consume events defined in the SAP SuccessFactors system:

- OAuth 2.0 client credentials grant flow.

This flow enables grant of an OAuth access token based on the client credentials only, without user interaction. See Configure Integration Based on OAuth 2.0 Client Credentials [page 1210]

- User propagation with SAML identity federation flow.
  This flow enables you to propagate the users from SAP SuccessFactors with SAML identity federation to your OAuth-protected extension application running on SAP BTP. See Configure Integration Based on User Propagation with SAML Identity Federation [page 1213]

5.2.2.10.1 Configure Integration Based on OAuth 2.0 Client Credentials

Use this procedure to enable your extension application to consume events defined in the SAP SuccessFactors system based on OAuth 2.0 client credentials grant.

Prerequisites

- You have an OAuth-protected extension application deployed in SAP BTP.
- You have specific Role-based permissions to use Intelligent Services Center. See Intelligent Services Center.
- You need to have specific Role-based permissions to use Integration Center. See Role-based Permissions for Integration Center.

Context

This integration flow enables grant of an OAuth access token based on the client credentials only, without user interaction.

Follow these steps to subscribe an extension application to receive notifications from an SAP SuccessFactors event of your choice:

> **i** Note
>
> Alternatively, in the Neo environment, you can perform **step 1** and **step 2** automatically, by configuring the `sfsf-outbound-connections` parameter in the Java application module of your Multi-Target Application (MTA) deployment descriptor. See MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment.

Procedure

1. Create OAuth Client in SAP BTP [page 1215].

2. Create Outbound OAuth Configuration in SAP SuccessFactors [page 1211].
3. Create a New Integration [page 1218].

5.2.2.10.1.1 Create OAuth Client in SAP BTP

Context

You have to create an OAuth client in SAP BTP with ID and secret. Later on, you will add these client ID and secret when creating the outbound OAuth configuration in SAP SuccessFactors system.

Procedure

1. Go to the SAP BTP cockpit and open the extension subaccount.
2. Choose *Security* > *OAuth*. Choose the *Clients* tab.
3. Choose *Register New Client*. The ID is automatically generated.
4. Enter a name for the client, for example, **`intsvcOAuthClient`**.
5. In the *Subscription* field, select the extension application that you want to be notified.
6. In the *Authorization Grant* field, select *Client Credentials*.
7. In the *Secret* field, define a password of your choice.
8. Choose *Save*.

5.2.2.10.1.2 Create Outbound OAuth Configuration in SAP SuccessFactors

Prerequisites

You have an OAuth client in SAP BTP created. See Create OAuth Client in SAP BTP [page 1215].

Context

Based on the OAuth client, you have to create an outbound OAuth configuration in the SAP SuccessFactors system.

Procedure

1. Log on to the SAP SuccessFactors system, and go to the *Integration Center*.
2. In the *Integration Center*, choose the *Security Center* tile and then choose the *Outbound OAuth Configurations*.
3. Choose *Add* to create a new outbound OAuth configuration.
4. In the *Configuration Name* field, enter a name for this outbound OAuth configuration.
5. In the *OAuth Type* drop-down menu, select *OAuth 2.0*.
6. In the *Grant Type* drop-down menu, select *Client_Credentials*.
7. In the *Client ID* field, paste the value of the *ID* field of the OAuth client that you have created in the SAP BTP cockpit.
8. In the *Client Secret* field, enter the value of the *Secret* field of the OAuth client that you have created in the SAP BTP cockpit.
9. In the *Token URL* field, paste the value of the Token URL from the *Token Endpoint* field in the *Security* > *OAuth* > *Branding* > *OAuth URLs*.
10. In the *Token Method* field, select *POST*.
11. Choose *Save*.

5.2.2.10.1.3 Create a New Integration

Prerequisites

- You have created an OAuth client in SAP BTP. See Create OAuth Client in SAP BTP [page 1215].
- You have created an outbound OAuth Configuration in SAP SuccessFactors. See Create Outbound OAuth Configuration in SAP SuccessFactors [page 1217].

Context

If you want your extension application to receive notifications from your SAP SuccessFactors system, you need to subscribe this application to a dedicated event. Using the Intelligent Services Center, you create and configure an integration for this specific event.

Procedure

1. Log on to the SAP SuccessFactors system, and go to the *Intelligent Services Center*.
2. Search for an event and choose it from the list. The event opens in a dedicated page.
3. Select an already existing flow or create a new one from the menu on the left.
4. In the *Activities* section on the right, choose *Integration*. A new dialog opens.
5. Choose *Create New Integration*.
    - For the *Destination Type*, choose *REST*. This means that the extension application that will receive notifications by this event is a REST service.
    - For the *Format*, choose *JSON*. This means that the notifications will be sent to the REST service in a JSON format.
6. Choose *Create*. The *Create New Integration Services* page appears.
7. In the *Create New Integration Services* page, in the *Options* tab, enter a name for the new integration and a description. Choose *Next*.
8. In the *Configure Fields* tab, you describe the JSON schema of the events which will be sent to the extension application REST API. Choose *Next*.
9. Configure the settings in the *Response Fields* and *Filter* tabs according to your scenario.
    - For the *Response Fields* tab, see Using Configure Fields Tab.
    - For the *Filter* tab, see Filter and Sort.
10. Choose *Next*.
11. In the *Destination Settings* tab, set the following configurations:
    1. In the *REST API URL* field, enter the REST endpoint URL of your application that will be called when the event notification is sent.
    2. In the *Authentication Type* field, select *OAuth*.
    3. In the *OAuth Configurations* field, select the outbound OAuth client credential s configuration you already have form the drop-down menu. See Create Outbound OAuth Configuration in SAP SuccessFactors [page 1211].
    4. Choose *Next*.
12. In the *Review and Run* tab, choose *Save* and then *Run Now*.

5.2.2.10.2 Configure Integration Based on User Propagation with SAML Identity Federation

Use this procedure to enable your extension application to consume events defined in the SAP SuccessFactors system based on user propagation with SAML identity federation.

Prerequisites

- You have an extension application deployed in SAP BTP.
- You have specific Role-based permissions to use Intelligent Services Center. See Intelligent Services Center.

- You need to have specific Role-based permissions to use Integration Center. See Role-based Permissions for Integration Center.

Context

This integration flow is based on exchanging the SAML (bearer) assertion from the SAP SuccessFactors identity provider for an OAuth access token from the SAP BTP authorization server. Using the access token, SAP SuccessFactors can access the OAuth-protected application.

You can integrate the extension application to subscribe to an event defined in the SAP SuccessFactors system. For example, if you want your extension application to be notified when a job title of an employee has been changed, you have to use the *Change in Job Title* event.

Follow these steps to subscribe an extension application to receive notifications from an SAP SuccessFactors event of your choice:

> **i** Note
>
> Alternatively, in the Neo environment, you can perform **step 1** to **step 4** automatically, by configuring the `sfsf-outbound-connections` parameter in the Java application module of your Multi-Target Application (MTA) deployment descriptor. See MTA Module Types, Resource Types, and Parameters for Applications in the Neo Environment.

Procedure

1. Generate OAuth X509 Key in SAP SuccessFactors [page 1214]
2. Create OAuth Client in SAP BTP [page 1215]
3. Create Trusted Identity Provider in SAP BTP Cockpit [page 1216]
4. Create Outbound OAuth Configuration in SAP SuccessFactors [page 1217]
5. Create a New Integration [page 1218]

5.2.2.10.2.1 Generate OAuth X509 Key in SAP SuccessFactors

Prerequisites

You need to have permissions for the *Integration Center* in the SAP SuccessFactors system.

Context

You have to generate the X.509 certificate in the SAP SuccessFactors system. Later on, you will add this certificate as a trusted identity provider in SAP BTP.

Procedure

1. Log on to the SAP SuccessFactors system, and go to the *Integration Center*.
2. In the *Integration Center*, choose the *Security Center* tile and then choose the *OAuth X509 Keys*.
3. Choose *Add* to create a new OAuth X509 key.
4. In the *Configuration Name*, enter a name of your choice for the OAuth X509 key.
5. In the *Common Name* field, replace the asterisk (*) with the SAP SuccessFactors company ID.
6. The *Description*, *Issued By*, *Organization*, *Organization Unit*, *Locality*, *State/Province*, and *Country*, *Valid for* fields are optional. You can fill them in as you like, there are no special requirements.
7. Choose *Regenerate and Save*. Choose *OK* when you are asked to confirm this operation.
8. Choose *Download X509 Certificate* and save the certificate on your local file system.

> **i** Note
>
> By default, the name of the certificate is *<configuration_name>_certificate.pem*.

5.2.2.10.2.2 Create OAuth Client in SAP BTP

Context

You have to create an OAuth client in SAP BTP with ID and secret. Later on, you will add these client ID and secret when creating the outbound OAuth configuration in SAP SuccessFactors system.

Procedure

1. Go to the SAP BTP cockpit and open the extension subaccount.
2. Choose *Security* > *OAuth*. Choose the *Clients* tab.
3. Choose *Register New Client*. The ID is automatically generated.
4. Enter a name for the client, for example, `intsvcOAuthClient`.
5. In the *Subscription* field, select the extension application that you want to be notified.

6. In the *Authorization Grant* field, select *Client Credentials*.
7. In the *Secret* field, define a password of your choice.
8. Choose *Save*.

5.2.2.10.2.3 Create Trusted Identity Provider in SAP BTP Cockpit

Prerequisites

You have created an OAuth X509 key and have saved the X509 certificate on your local file system. See Generate OAuth X509 Key in SAP SuccessFactors [page 1214].

Context

Register the certificate you downloaded when generating the OAuth X509 key in the SAP BTP cockpit.

Procedure

1. Go to the SAP BTP cockpit.
2. Choose *Security* > *Trust* > *Application Identity Provider*.
3. Choose *Add Trusted Identity Provider*.
4. In the *Name* field, enter a name for the trusted identity provider.
5. Find the certificate saved on your local file system and open it in a text editor. This is the certificate you downloaded when you created the OAuth X509 key. See Generate OAuth X509 Key in SAP SuccessFactors [page 1214].
6. Copy the certificate content and paste it in the *Sigining Certificate* field.
7. Select the *Only for IDP-Initiated SSO* checkbox.

   > **i Note**
   >
   > You don't need to change anything in the rest of the fields.

8. Choose *Save*.

5.2.2.10.2.4 Create Outbound OAuth Configuration in SAP SuccessFactors

Prerequisites

- You have an OAuth client in SAP BTP created. See Create OAuth Client in SAP BTP [page 1215].
- You have created an OAuth X509 key and have saved the X509 certificate on your local file system. See Generate OAuth X509 Key in SAP SuccessFactors [page 1214].
- You have registered the X509 certificate when creating a trusted identity provider in the SAP BTP cockpit. See Create Trusted Identity Provider in SAP BTP Cockpit [page 1216].

Context

Based on the X509 certificate and OAuth client, you have to create an outbound OAuth configuration in the SAP SuccessFactors system.

Procedure

1. Log on to the SAP SuccessFactors system, and go to the *Integration Center*.
2. In the *Integration Center*, choose the *Security Center* tile and then choose the *Outbound OAuth Configurations*.
3. Choose *Add* to create a new outbound OAuth configuration.
4. In the *Configuration Name* field, enter a name for this outbound OAuth configuration.
5. In the *Client ID* field, paste the value of the *ID* field of the OAuth client that you have created in the SAP BTP cockpit.
6. In the *Client Secret* field, enter the value of the *Secret* field of the OAuth client that you have created in the SAP BTP cockpit.
7. In the *OAuth Type* drop-down menu, select *OAuth 2.0 with SAML Flow*.
8. In the *Token URL* field, paste the value of the Token URL from the *Token Endpoint* field in the *Security* > *OAuth* > *Branding* > *OAuth URLs*.
9. In the *Token Method* field, select *POST*.
10. In the *Audience* field, paste the local service provider name for your account from the SAP BTP cockpit.

> **i Note**
>
> Open the SAP BTP cockpit and go to *Security* > *Trust* > *Local Service Provider* > *Local Provider Name*. See Principal Propagation to OAuth-Protected Applications.

11. In the *Recipient* field, enter the same value as the one in the *Token URL* field.
12. In the *Issuer* field, copy and paste the same value as the *Name* field of the trusted identity provider you created in the SAP BTP cockpit. See Create Trusted Identity Provider in SAP BTP Cockpit [page 1216].
13. In the *X509 Keys*, select the OAuth X509 key you created. See Generate OAuth X509 Key in SAP SuccessFactors [page 1214].
14. Choose *Save*.

5.2.2.10.2.5 Create a New Integration

Prerequisites

- You have created an OAuth X509 key and have saved the X509 certificate on your local file system. See Generate OAuth X509 Key in SAP SuccessFactors [page 1214].
- You have created an OAuth client in SAP BTP. See Create OAuth Client in SAP BTP [page 1215].
- You have created a trusted identity provider in SAP BTP. See Create Trusted Identity Provider in SAP BTP Cockpit [page 1216].
- You have created an outbound OAuth Configuration in SAP SuccessFactors. See Create Outbound OAuth Configuration in SAP SuccessFactors [page 1217].

Context

If you want your extension application to receive notifications from your SAP SuccessFactors system, you need to subscribe this application to a dedicated event. Using the Intelligent Services Center, you create and configure an integration for this specific event.

Procedure

1. Log on to the SAP SuccessFactors system, and go to the *Intelligent Services Center*.
2. Search for an event and choose it from the list. The event opens in a dedicated page.
3. Select an already existing flow or create a new one from the menu on the left.
4. In the *Activities* section on the right, choose *Integration*. A new dialog opens.
5. Choose *Create New Integration*.
    - For the *Destination Type*, choose *REST*. This means that the extension application that will receive notifications by this event is a REST service.
    - For the *Format*, choose *JSON*. This means that the notifications will be sent to the REST service in a JSON format.

6. Choose *Create*. The *Create New Integration Services* page appears.
7. In the *Create New Integration Services* page, in the *Options* tab, enter a name for the new integration and a description. Choose *Next*.
8. In the *Configure Fields* tab, you describe the JSON schema of the events which will be sent to the extension application REST API. Choose *Next*.
9. Configure the settings in the *Response Fields* and *Filter* tabs according to your scenario.
    - For the *Response Fields* tab, see Using Configure Fields Tab.
    - For the *Filter* tab, see Filter and Sort.
10. Choose *Next*.
11. In the *Destination Settings* tab, set the following configurations:
    1. In the *REST API URL* field, enter the REST endpoint URL of your application that will be called when the event notification is sent.
    2. In the *Authentication Type* field, select *OAuth*.
    3. In the *OAuth Configurations* field, select the outbound OAuth configuration you already have form the drop-down menu. See Create Outbound OAuth Configuration in SAP SuccessFactors [page 1217].
    4. In the *OAuth 2.0 with SAML Flow Options* section, in the *SAML Subject NameID Path* field, choose *Browse*.
    5. Choose the *Associate* field. Expand the *Entity Tree View* and select the field that will provide the value for the SAML Subject NameID. Choose *Change Association to Selected Field*.
    6. Choose *Next*.
12. In the *Review and Run* tab, choose *Save* and then *Run Now*.

5.3 Extending SAP Cloud for Customer in the Neo Environment

This section guides you through the configuration tasks that you need to perform to enable the Neo environment for developing extension applications for your SAP Cloud for Customer system.

Overview

Extending SAP Cloud for Customer on SAP BTP allows you to implement additional workflows or modules on top of the SAP Cloud for Customer benefiting from out-of-the-box security, inherited data access governance, user interface embedding, and others.

In the SAP Cloud for Customer extensions scenarios, these are the important aspects:

- The Extension Applications for SAP Cloud for Customer are hosted or subscribed in a dedicated subaccount in SAP BTP to ensure the consistency in the integration configuration between the two solutions. The purpose of the subaccount is to hold the common integration configurations for all extension applications.
- The single sign-on configuration between SAP Cloud for Customer and SAP BTP ensures the secure and consistent data access for the extension application.

- OAuth connectivity configuration for enabling the use of SAP Cloud for Customer OData APIs.
- Configuration of the HTML mashups in SAP Cloud for Customer helps with embedding the extension application UI directly in the SAP Cloud for Customer screens and offers the same look and feel to the end users.

Prerequisites

- Your company has an enterprise account in the cloud platform. For more information about account types and purchasing an enterprise account, see:
  - Account Model
  - Get an Enterprise Account
- You have a user account on the platform with administrative permissions for the enterprise account of your company and you can use the Cloud Cockpit.
- If you need to configure single sign-on using the Identity Authentication service, you have to make sure that your company has license and a tenant for this SAP service. For more details, see the Getting Started section of the documentation for this service.
- You have administrative permissions for the SAP Cloud for Customer system necessary for configuring the connectivity with external systems, in this case SAP BTP.

Process Flow

You have to configure the cloud platform extension integration with SAP Cloud for Customer to enable the use of applications running on top of the platform from SAP Cloud for Customer.

1. Authentication and Principal Propagation

A single sign-on configuration between SAP Cloud for Customer and SAP BTP and the principal propagation enablement ensure the secure and consistent access for the extension solutions. If your solution requires system-to-system data access, with no user propagation, you can implement it using a user with Basic authentication and no principal propagation.

> → Tip
>
> We recommend that you use SSO and OData access with principal propagation, to ensure that the data is accessed on behalf of the proper authorized user.

Implementation with SSO and principal propagation

When your scenario requires SSO and principal propagation, the SAP Cloud for Customer system and the SAP BTP subaccount (where the extension application is deployed or subscribed) have to trust each other and use one and the same identity provider. For more information, see Configuring Single Sign-On [page 1221].

Implementation without SSO and principal propagation

If your scenario does not require SSO, principal propagation, or UI integration, you use a dedicated user and create an HTTP destination with Basic Authentication to configure the connectivity. For more information, see Create and Configure the HTTP Destination [page 1237].

2. Connectivity Between SAP BTP and SAP Cloud for Customer

When your extension application needs to do read and write operations from/to the SAP Cloud for Customer system, you have to configure the connectivity to enable the use of SAP Cloud for Customer OData APIs. You need such connectivity to be established also when you want to embed the UIs of the extension solution into the SAP Cloud for Customer user interface.

The connectivity configuration steps are required on both systems and can be implemented with the following options depending on the security mechanisms for protecting the connectivity:

Using OAuth

> **i** Note
>
> OAuth can be used only when a common identity provider is configured for both systems.

- On SAP Cloud for Customer side:
  - Configure OAuth Identity Provider in SAP Cloud for Customer [page 1235]
  - Configure the OAuth Client for OData Access [page 1236]
- On the SAP BTP side:
  - Create an HTTP destination with OAuthSAMLBearerAssertion for authentication method. For more details, see Create and Configure the HTTP Destination [page 1237].

Using Basic Authentication

- On SAP Cloud for Customer side:
  - (If not available) Create a technical user with permissions to access the required SAP Cloud for Customer OData APIs and use it for the HTTP destination configuration when you choose Basic authentication for protecting the connectivity.
- On the SAP BTP side:
  - Create an HTTP destination with Basic authentication using the credentials of a technical user in the SAP Cloud for Customer system. For more details, see Create and Configure the HTTP Destination [page 1237].

3. User Interface Integration

The extension capabilities of SAP BTP allow developers to embed the user interface of the new solution in the SAP Cloud for Customer screens and this way to offer seamless end-user experience.

5.3.1 Configuring Single Sign-On

You can configure the cloud platform extension integration with SAP Cloud for Customer to enable the use of applications running on top of the platform from SAP Cloud for Customer.

Using Identity Authentication

Identity Authentication service provides authentication, single sign-on, and on-premise integration. Identity Authentication service is closely integrated with SAP BTP, and it is offered as part of the platform.

To ensure the required security for accessing the extension applications, you need to configure the Single Sign-On between the SAP BTP extension subaccount and the SAP Cloud for Customer tenant using a SAML identity provider, for example Identity Authentication. The Single Sign-On requires both solutions to be configured as trusted SAML service providers for the Identity Authentication service and on the other side the Identity Authentication service to be configured as trusted identity provider for the two solutions.

In this scenario, the authentication to SAP Cloud for Customer extension applications is restricted to the authorized users. The identity of a user is verified by the identity provider, as specified by SAML 2.0. The identity provider, (Identity Authentication), stores a list of all users that are allowed to access the service provider (SAP Cloud for Customer) along with their credentials. The integration between the SAP Cloud for Customer and the Identity Authentication is based on trust configuration. When a user attempts to access SAP Cloud for Customer for the first time, the system redirects the user to the identity provider for identification. From then on, the user session is kept active, and the user is no longer prompted for credentials when he or she, for example, tries to use the extension application. This is called Single Sign-On (SSO).

Using Third-Party Identity Provider

You can use a third-party identity provider (which means a different from Identity Authentication) as well to ensure the required security for your landscape. In this case you also need to perform a few configuration tasks on all the sides - SAP BTP, SAP Cloud for Customer, and the identity provider that you are using.

Related Information

Configuring Single Sign-On Using Identity Authentication [page 1222]
Configuring Single Sign-On Using Third-Party Identity Provider [page 1234]

5.3.1.1 Configuring Single Sign-On Using Identity Authentication

To ensure the required security for your landscape you need to perform a few configuration tasks on all the sides - SAP BTP, Identity Authentication and SAP Cloud for Customer.

Context

The following procedure describes how to configure Identity Authentication and SAP Cloud for Customer to use the authentication and single sign-on capabilities based on the industry standard SAML 2.0. You can also find information about how to configure the trust between Identity Authentication and the SAP BTP subaccount, in case it is not configured by default.

Procedure

1. Configure the trust settings between Identity Authentication and your SAP Cloud for Customer system. For more information, see Setting Up Trust Between Identity Authentication and SAP Cloud for Customer [page 1223].

   > **i** Note
   >
   > If you already have a SAP Cloud for Customer system with existing users, you need to define these users in the Identity Authentication service as well. To do that, export as a CSV file the users of your company employees from your SAP Cloud for Customer system and import it into Identity Authentication.

2. Configure the SAP BTP trust settings and add the tenant of Identity Authentication available for your company as a SAML identity provider. For more information, see: Setting Up Trust Between Identity Authentication and SAP BTP [page 1232].

   > **i** Note
   >
   > Use **Step 2** of this procedure only if the trust between the SAP BTP subaccount and Identity Authentication has not been automatically configured after you have purchased a subscription for your Identity Authentication tenant.

5.3.1.1.1 Setting Up Trust Between Identity Authentication and SAP Cloud for Customer

To use Identity Authentication as a common identity provider between SAP BTP and SAP Cloud for Customer, first you need to configure it for the SAP Cloud for Customer system.

Context

Procedure

1. Configuring Identity Authentication [page 1224].
2. Configuring SAP Cloud for Customer [page 1225].
3. Configuring the Application's Home URL [page 1226].

5.3.1.1.1.1 Configuring Identity Authentication

Use this procedure to configure the service provider (SAP Cloud for Customer) in the Identity Authentication tenant and to define the identity federation.

Context

Procedure

1. Get the service provider metadata XML file. To do so, you download the metadata XML file of your SAP Cloud for Customer system:
   a. Log on to your SAP Cloud for Customer system as an administrator.
   b. Select *ADMINITRATOR* > *Common Tasks* and then choose *Configure Single Sign-On*.
   c. On the *CONFIGURE SINGLE SIGN-ON* screen, navigate to *MY SYSTEM* > *GENERAL* > *Download Metadata*.
   d. Choose the *SP Metadata* link, to download the SAP Cloud for Customer metadata XML file. Save it locally on your file system to import in later on in theIdentity Authentication.
2. Create a custom application to use it as a SAML2.0 service provider. Access the tenant's administration console for Identity Authentication by using the console's URL.

   The URL has the `https://<tenant ID>.accounts.ondemand.com/admin` pattern.

   You can also get the URL from the Identity Authentication tenant registration e-mail.

   a. Choose *Applications & Resources* > *Applications* from the menu on the left.
   b. Choose the *+Add* button on the left hand panel, and enter the name of your SAP Cloud for Customer system.

      > **i Note**
      >
      > The name of the application is displayed on the login and registration pages.

   c. Choose *Save*.

   The system creates an application and adds it to the list of applications.
3. Configure the SAML 2.0 trust with SAP Cloud for Customer as a service provider. In the tenant's administration console for Identity Authentication, execute the following steps:
   a. On the left-hand side, choose the newly created application, and then choose *Trust*.
   b. Choose *SAML 2.0 Configuration*.
   c. Upload the metadata XML file of your SAP Cloud for Customer system that you have downloaded in **Step 1**.

      On service provider metadata upload, the fields are populated with the parsed data from the XML file.
   d. Save the configuration settings.

4. Configure the identity federation on Identity Authentication. To do so, proceed as follows:
   a. In the tenant's administration console for Identity Authentication, choose your SAP Cloud for Customer application, and navigate to *Trust* > *Name ID Attribute*, and then select *Login Name*.

      This is the profile attribute that Identity Authentication sends to the application as a name ID. The application then uses this attribute to identify the user. You should select the attribute expected by your SAP Cloud for Customer system as a valid user.

      > **i Note**
      >
      > E-mail is not supported by the SAP Cloud for Customer system.

   b. Save your selection.

5.3.1.1.1.2 Configuring SAP Cloud for Customer

Configure the Single Sign-On (SSO) to Identity Authentication in the SAP Cloud for Customer system.

Context

Procedure

1. Get the identity provider metadata XML file. To do so:
   a. Use the following URL to access the metadata for the Identity Authentication tenant:

      `https://<tenant ID>.accounts.ondemand.com/saml2/metadata`.
   b. Save the content of the page locally on your file system as an XML file.
2. Log on to your SAP Cloud for Customer system as an administrator.
3. Choose *ADMINISTRATOR* > *Common Tasks* and then choose *Configure Single Sign-On*.
4. On the *CONFIGURE SINGLE SIGN-ON* screen, choose the *IDENTITY PROVIDER* tab.
5. Choose *New Identity Provider*.
6. Browse and open the metadata XML file that you have downloaded in **Step 1**. By importing the metadata, the system automatically uploads the required signature certificate and encryption certificate.

   The new identity provider is activated and displayed in the *Trusted Identity Provider* list.
7. Once you have configured your identity provider, activate SSO in your SAP Cloud for Customer system. To do so, choose *Activate Single Sign-On*, and then choose *OK*.
8. To save your settings, choose *Save* in the upper left-hand corner.

5.3.1.1.1.3 Configuring the Application's Home URL

You can configure the *Home URL* of an application in the administration console for Identity Authentication.

Context

Home URL is the URL that the user is redirected to after being authenticated. Initially, the *Home URL* for an application is not configured in the administration console for Identity Authentication. Once the URL has been set, you can change it.

> → Remember
>
> *Home URL* is necessary when you import new users in Identity Authentication. Identity Authentication needs to send activation e-mails to the new users and the home URL has to be mentioned in the e-mails. To access the application, the users have to activate their user accounts. For more information, see Importing or Updating SAP Cloud for Customer Users in Identity Authentication [page 1229].

To configure the *Home URL*, proceed as follows:

Procedure

1. Access the tenant's administration console for Identity Authentication by using the console's URL.

   > i Note
   >
   > The URL has the `https://<tenant ID>.accounts.ondemand.com/admin` pattern.
   >
   > *Tenant ID* is an automatically generated ID by the system. The first administrator created for the tenant receives an activation e-mail with a URL in it. This URL contains the *tenant ID*.

2. Choose *Applications & Resources* > *Applications* from the menu on the left.

   This operation opens a list of the applications.

3. Choose the application that you want to edit.

   > i Note
   >
   > Type the name of the application in the search field to filter the list items, or choose the application from the list on the left.
   >
   > If you do not have a created application in your list, you can create one. For more information, see Create a New Application.

4. Select the *Home URL* anchor text.

> **Note**
>
> You get the Home URL from the SAP Cloud for Customer system:
>
> - Log on to your SAP Cloud for Customer system as an administrator.
> - Choose *ADMINITRATOR* > *Common Tasks* and then choose *Configure Single Sign-On*.
> - In the *Single Sign-On URL Handling* section, copy and save the URL from the *SSO URL* field. If this URL starts with uppercase *HTTPS* protocol, replace it with **`https`**.

   - If you are configuring the URL for the first time, type the address in the pop-up dialog that appears.
   - If you are editing the URL, choose *Edit* from the list item, and type the new address in the pop-up dialog.

5. Save your changes.

   Once the application has been updated, the system displays the message `Application <name of application> updated`.

5.3.1.1.2 Uploading Users to Identity Authentication

When you configure the Identity Authentication service as the SAML identity provider for your SAP Cloud for Customer system, you have to make sure that all users from SAP Cloud for Customer will have an identity record in the Identity Authentication service. For this purpose, you have to export the user details in a CSV file format and then use this CSV file to import these users into the Identity Authentication tenant of your company that will be used as the identity provider.

Context

To upload the users of your SAP Cloud for Customer company into Identity Authentication, you export the user base and import it into Identity Authentication in CSV format.

Procedure

1. Export the SAP Cloud for Customer users and save the data in CSV format. For more information, see Exporting SAP Cloud for Customer Users [page 1228]
2. Import the users (user names and user IDs) in the Identity Authentication tenant with the CSV file you have exported from your SAP Cloud for Customer system. For more information, see Importing or Updating SAP Cloud for Customer Users in Identity Authentication [page 1229].

5.3.1.1.2.1 Exporting SAP Cloud for Customer Users

Use this procedure to export SAP Cloud for Customer users and save the user data in CSV format.

Context

You need to export the users from the SAP Cloud for Customer system, adapt the data and save it in CSV format so that you can import the data in the Identity Authentication tenant and thus allow your company employees to authenticate with their corporate credentials.

> **i** Note
>
> It is mandatory to have an e-mail for every user that you replicate from the SAP Cloud for Customer system in your SAP BTP subaccount.
>
> Although in the SAP Cloud for Customer system having an e-mail for a user is not obligatory, you need to have this e-mail as a parameter when you export the users and import them in your SAP BTP subaccount.

The CSV file with the SAP Cloud for Customer user data must contain the following columns:

- `status`
- `loginName`
- `mail`
- `firstName`
- `lastName`

Procedure

1. Log on to your SAP Cloud for Customer system as an administrator.
2. Open the Business Users view. To do so, choose *ADMINISTRATOR* > *GENERAL SETTINGS* > *Business Users*.
3. To export the users, choose *Export* > *To Microsoft Excel®*.

   The system exports the users in a Microsoft Excel® spreadsheet. The spreadsheet contains the following columns: *User Locked*, *Password Locked*, *User ID*, *Name*, *Technical ID*. The spreadsheet does not contain a column with the user e-mails and the user status.
4. Modify the spreadsheet as follows:
   a. Remove the superfluous data and leave only the *User ID* and *Name* columns.
   b. Rename the *User ID* column to **`loginName`** and split the *Name* column to **`firstName`** and **`lastName`**columns.

      The data in the *loginName* stays the same, while the names from the former *Name* columns are split in the following way: first names go to the *firstName* column and last names go to the *lastName* column.

c. Insert a new column before the *loginName* column and name it **`status`**. The status can be **`active`** or **`inactive`**.

d. Insert a new column between *loginName* and *firstName* and name it **`mail`**.

   To get the e-mail of a user, select the user. The e-mail is displayed on the *General Data* tab page.

> **i Note**
>
> - The *status*, *loginName*, *mail* and *firstName* columns must be with a string value of up to 32 characters. The *lastName* column must be with a string value of up to 64 characters.
> - The names in the *mail* and *loginName* columns must be unique.
> - You cannot change the e-mail of an existing user.
> - The *status* column defines whether the user is still active in the system and is able to work with any tenant applications. When a user is deleted, it is rendered inactive.

e. Save the file in CSV format.

   After you have modified and saved your file, you should have a CSV file with the following columns and similar data.

status	loginName	mail	firstName	lastName
active	EID00001	michael.adams@example.com	Michael	Adams
active	EID00002	julie.armstrong@example.com	Julie	Armstrong
active	EID00003	donna.moore@example.com	Donna	Moore

5.3.1.1.2.2 Importing or Updating SAP Cloud for Customer Users in Identity Authentication

As a tenant administrator of the Identity Authentication, you can import new users or update existing ones for a specific application with a CSV file, and send activtation e-mails to the users that have not received activation e-mails for that application so far.

Prerequisites

- You are assigned the *Manage Applications* and *Manage Users* roles. For more information about how to assign administrator roles, see Edit Administrator Authorizations.
- You have configured the trust between Identity Authentication tenant and the SAP Cloud for Customer system. For more information see Setting Up Trust Between Identity Authentication and SAP Cloud for Customer [page 1223].

> **Note**
>
> You need the metadata to configure the trust between the service provider and Identity Authentication, which is in the role of identity provider.

Context

By importing new users with a CSV file, you create user profiles without passwords in Identity Authentication. As a result, the users receive e-mails with instructions how to activate their user accounts. After the users set their passwords, they can log on to the application for which they were imported. Based on the user access configuration of the application, the users can log on to other applications connected with the tenant in Identity Authentication.

> **Note**
>
> When a new application is created in the Identity Authentication tenant, the default value for user access is internal and you have to keep it like this. If you decide to change the user access to private for your SAP BTP subaccount or for the SAP Cloud for Customer system, you have to make sure you import the users into that application.

In addition to the new user import, you can specify existing users in the imported CSV file. You thus define the users to be updated in Identity Authentication.

The CSV file contains these columns *status*, *loginName*, *mail*, *firstName*, *lastName*. These columns are mandatory and they must always have values.

The *status*, *loginName*, *mail* and *firstName* columns must be with a string value of up to 32 characters. The *lastName* column must be with a string value of up to 64 characters.

The names in the *mail* and *loginName* columns must be unique.

> **Caution**
>
> You cannot change the e-mail of an existing user.

The *status* column defines whether the user is still active in the system and is able to work with any tenant applications. When a user is deleted, it is rendered inactive.

> **Example**

status	loginName	mail	firstName	lastName
active	EID00001	michael.adams@example.com	Michael	Adams
active	EID00002	julie.armstrong@example.com	Julie	Armstrong
active	EID00003	donna.moore@example.com	Donna	Moore

To import users for an application into Identity Authentication, and to send activation e-mails, proceed as follows:

Procedure

1. Access the tenant's administration console for Identity Authentication by using the console's URL.

   > **i Note**
   >
   > The URL has the `https://<tenant ID>.subaccounts.ondemand.com/admin` pattern.
   >
   > *Tenant ID* is an automatically generated ID by the system. The first administrator created for the tenant receives an activation e-mail with a URL in it. This URL contains the *tenant ID*.

2. Choose the *Import Users* tile.

   This operation opens the *Import Users* page.

3. Choose the application that you want to edit.

   > **i Note**
   >
   > Type the name of the application in the search field to filter the list items, or choose the application from the list on the left.
   >
   > If you do not have a created application in your list, you can create one. For more information, see Configuring Identity Authentication [page 1224].

4. Choose the *Browse...* button and specify the location of the CSV file.

   > **i Note**
   >
   > Use a file smaller than 100 KB and with an extension `.csv`. If your file is 100 KB or larger, you have to import the user information in iterations with smaller size files.

5. Choose the *Import* button.

   If the operation is successful, the system displays the message `Users imported or updated`.

6. Choose the one of the following options:

Option	Description
Do nothing	The users are imported or updated for the selected application, but they will not receive activation e-mails. The activation e-mails will be sent when you choose *Send E-Mails* > *Send*.
Repeat steps 2 to 5	The users are imported or updated for the selected application, but they will not receive activation e-mails. The activation e-mails will be sent when you choose *Send E-Mails* > *Send*.
Choose *Send E-Mails* > *Send*	This will send activation e-mails to all users that are imported for the selected application, but have not received activation e-mails so far.

Option	Description
	i Note The *Send* button is inactive if *Home URL* or SAML 2.0 configuration of the application is missing. You can only import users, but you cannot send activation emails. You need the *Home URL* configured for the specific application to be able to send the activation e-mails to the imported new users. For more information, see Configuring the Application's Home URL [page 1226]. To access the application, the users have to activate their user accounts by following the link they receive in the e-mails.

5.3.1.1.3 Setting Up Trust Between Identity Authentication and SAP BTP

Use this procedure to configure the SAP BTP trust settings and to add the tenant of Identity Authentication registered for your current SAP user as an identity provider.

Context

Identity Authentication is closely integrated with SAP BTP, and it is offered as part of most of the cloud platform packages. For those packages the trust between the subaccount in SAP BTP and Identity Authentication is configured automatically and the tenant for Identity Authentication is set up by default, once you have a partner or customer subaccount. However, you can manually configure the trust and set up the Identity Authentication tenant if your scenario requires it.

Procedure

1. Open the SAP BTP cockpit, select the region in which your subaccount is hosted, then select the global account that contains your subaccount, and then choose the tile of your subaccount. For more information about the regions, see Regions and Hosts.
2. Choose *Security* > *Trust* in the navigation.
3. Choose the *Local Service Provider* tab, and then choose *Edit*.
4. From the *Configuration Type* dropdown box, select *Custom*.
5. Configure the signing key and the signing certificate as follows:
   - If you want to use a signing key and a self-signed certificate automatically generated by the system, choose *Generate Key Pair*.
   - If you have your own key and certificate generated from an external application and signed by a trusted CA), you can use them instead of using the ones generated by the SAP BTP. To do so, copy the

Base64-encoded signing key in the *Signing Key* field, and then copy the textual content of the certificate in the *Signing Certificate* field.

6. From the *Principal Propagation* dropdown box, select *Enabled*. Thus, you enable the propagation of the principal ID in the OAuth destination and hence, the application-to-application Single Sign-On (SSO).
7. Choose *Save*.
8. Choose the *Get Metadata* link to download the SAP BTP metadata for your subaccount. You will need this metadata in **Step 17**.
9. Save the metadata of your Identity Authentication tenant on your local file system as an XML file. You can find the tenant at `https://<tenant ID>.accounts.ondemand.com/saml2/metadata`. You will need this metadata in **Step 10**.
10. Go to SAP BTP cockpit and choose *Security* > *Trust* > *Application Identity Provider*. Then choose *Add Trusted Identity Provider*. In the *General* tab, upload the Identity Authentication metadata XML file (from **Step 9**) in the *Metadata File* field.
11. In the *Attributes* tab, choose *Add Assertion-Based Attribute* to add the following attribute mappings (after adding one pair, choose *Add Assertion-Based Attribute* again to add more input fields).

Assertion Attribute	Principle Attribute
first_name	firstname
last_name	lastname
mail	email

12. Choose *Save*.
13. Access the tenant's administration console for Identity Authentication by using the console's URL.

    The URL has the `https://<tenant ID>.accounts.ondemand.com/admin` pattern.

    You can also get the URL from the Identity Authentication tenant registration e-mail.
14. Choose the *Applications & Resources* > *Applications*.
15. Choose the *+Add* button on the left hand panel, and enter the name of your subaccount in SAP BTP.
16. Choose *Save*.
17. Configure the SAML 2.0 trust with your subaccount in SAP BTP as a service provider. To do so, proceed as follows:
    a. On the left-hand side, choose the newly created application, and then choose *Trust*.
    b. Choose *SAML 2.0 Configuration*.
    c. Upload the metadata XML file of your subaccount in SAP BTP that you have downloaded in **Step 8**.

       On service provider metadata upload, the fields are populated with the parsed data from the XML file.
    d. Save the configuration settings.
18. Configure the identity federation on Identity Authentication. To do so, proceed as follows:
    a. You are still in the tenant's administration console for Identity Authentication, choose *Name ID Attribute* and then select *Login Name*.

       This is the profile attribute that Identity Authentication sends to the application as a name ID. The application then uses this attribute to identify the user.
    b. Save your selection.

Results

The trust will be established automatically upon registration on both the SAP BTP and Identity Authentication tenant side.

Related Information

Get Started with Identity Authentication
ID Federation with a Identity Authentication Tenant

5.3.1.2 Configuring Single Sign-On Using Third-Party Identity Provider

To ensure the required security for your landscape you need to perform a few configuration tasks on all the sides - SAP BTP, SAP Cloud for Customer, and the identity provider that you are using (if this provider is different from Identity Authentication, for which there is a dedicated section).

Context

Procedure

1. Set up trust between your identity provider and SAP Cloud for Customer system.

   For more information, see section **Configure Your Solution for Single Sign-On** in the SAP Cloud for Customer Security Guide.
2. Set up trust between your identity provider and SAP BTP.

   For more information, see Application Identity Provider [page 1720].

5.3.2 Configuring the SAP BTP Application Connectivity to SAP Cloud for Customer APIs

You configure the connectivity to enable the use of SAP Cloud for Customer OData APIs.

- Create and configure the OAuth client for OData access to enable the connectivity to SAP Cloud for Customer OData APIs. For more information, see Configure the OAuth Client for OData Access [page 1236].

- Configure the HTTP destination required to create an HTTP client for the OData API. For more information, see Create and Configure the HTTP Destination [page 1237].

When your extension application needs to do read and write operations from/to the SAP Cloud for Customer system, you have to configure the connectivity to enable the use of SAP Cloud for Customer OData APIs. You need such connectivity to be established also when you want to embed the UIs of the extension solution into the SAP Cloud for Customer user interface.

The connectivity configuration steps are required on both systems and can be implemented with the following options depending on the security mechanisms for protecting the connectivity:

Using OAuth

> **i** Note
>
> OAuth can be used only when a common identity provider is configured for both systems.

- On SAP Cloud for Customer side:
  - Configure OAuth Identity Provider in SAP Cloud for Customer [page 1235]
  - Configure the OAuth Client for OData Access [page 1236]
- On the SAP BTP side:
  - Create an HTTP destination with OAuthSAMLBearerAssertion for authentication method. For more details, see Create and Configure the HTTP Destination [page 1237].

Using Basic Authentication

- On SAP Cloud for Customer side:
  - (If not available) Create a technical user with permissions to access the required SAP Cloud for Customer OData APIs and use it for the HTTP destination configuration when you choose Basic authentication for protecting the connectivity.
- On the SAP BTP side:
  - Create an HTTP destination with Basic authentication using the credentials of a technical user in the SAP Cloud for Customer system. For more details, see Create and Configure the HTTP Destination [page 1237].

5.3.2.1 Configure OAuth Identity Provider in SAP Cloud for Customer

You need to add the SAP BTP service provider as a trusted OAuth identity provider.

Context

Procedure

1. In the SAP BTP cockpit, open the trust management setting of the subaccount. To do so, log on to the SAP BTP cockpit, select the region in which your subaccount is hosted, then select the global account that contains your subaccount, and then choose the tile of your subaccount. Choose *Security* > *Trust*.
   a. On the *Local Service Provider* tab page, choose *Edit* and select *Custom* from the dropdown list of the *Configuration Type* field.
   b. Copy and save the entry from the *Local Provider Name* field.
   c. Copy the entry from the *Signing Certificate* field, and save it in a file with the following format `<subaccount>_signing.cer`, where `<subaccount>` is the *Subaccount Name* from the *Subaccount Information* of your SAP BTP subaccount.
2. Log on to your SAP Cloud for Customer system as an administrator. Go to *ADMINISTRATOR* > *Common Tasks*. Choose *Configure OAuth 2.0 Identity Provider* > *New OAuth2.0 Provider*, and configure the settings as follows:
   - In the *Issuing Entity Name* field, paste the entry that you copied on **step 1b** (the entry from the *Local Provider Name* field in the trust managing settings of the SAP BTP subaccount).
   - From the *Primary Signing Certificate* field, browse the `<subaccount>_signing.cer` file that you saved on **step 1c**.
   - Select the *E-Mail Address* checkbox.
3. Choose *Submit*.

5.3.2.2 Configure the OAuth Client for OData Access

In SAP Cloud for Customer, use this procedure to configure the OAuth client for OData access to SAP Cloud for Customer OData APIs.

Context

Procedure

1. Log on to your SAP Cloud for Customer system as an administrator. Go to *ADMINISTRATOR* > *OAUTH2.0 CLIENT REGISTRATION*.
2. Choose *New*.
3. Specify a client secret, description, and token lifetime (in seconds). For example, 3600 seconds.
4. In the *Issuer Name* field, use the dropdown list to specify the identity provider you are using:
   - if you have triggered an automated integration between SAP Cloud for Customer and SAP BTP, the identity provider was registered during this automated integration. To check which is this identity

provider, go to SAP BTP cockpit and then choose *Security* > *Trust* > *Local Service Provider*. The value of the *Local Service Provider* field is the identity provider you need to select from the dropdown list.

- if you don't use an automated integration, use the identity provider that you created in the document Configure OAuth Identity Provider in SAP Cloud for Customer [page 1235].

5. Copy the entry in the *Client ID* field. You will need it later when creating the HTTP destination with OAuth authentication required for the connectivity to the SAP Cloud for Customer OData APIs.
6. In the *Scope* list, select the *UIWC:CC_HOME* scope.
7. Choose *Save and Close*.

Next Steps

On the SAP BTP side configure the HTTP destination required to create an HTTP client for the OData API, and thus ensure the connectivity toSAP Cloud for Customer. For more information, see Create and Configure the HTTP Destination [page 1237].

Related Information

Configuring OAuth 2.0

5.3.2.3 Create and Configure the HTTP Destination

Use this procedure to configure the HTTP destination in the SAP BTP subaccount required to create an HTTP client for the OData API.

Prerequisites

- You have configured the OAuth client for OData access. For more information, see Configure the OAuth Client for OData Access [page 1236].
- You have logged into the SAP BTP cockpit from the SAP BTP landing page for your subaccount.

Context

You create and configure the destinations on subaccount or application level in the SAP BTP cockpit.

Depending on your scenario, you use either OAuth or basic authentication for accessing the SAP Cloud for Customer system and the extension applications. If your scenario requires the SAP Cloud for Customer system

and the extension applications to support Single Sign-On, you create and configure an HTTP destination with OAuth authentication. Otherwise, you can use an HTTP destination with basic authentication.

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose *Connectivity* > *Destinations* in the navigation panel.
3. Depending on your scenario, create an HTTP destination as follows:
   - To enable the support of SSO, create an *OAuth2SAMLBearerAssertion* HTTP destination and configure its settings as follows:
     1. Configure the basic settings:

Parameter	Value
`Name`	Enter a meaningful name.
`Type`	**`HTTP`**
`Description`	(Optional) Enter a meaningful description.
`URL`	**`https://<my_SAP_Cloud_for_Customer_system_name>.crm.ondemand.com/sap/c4c/odata/v1/c4codata`**
`Proxy Type`	**`Internet`**
`Authentication`	**`OAuth2SAMLBearerAssertion`**
`Audience`	Take this value from the *Local Service Provider* field in *Configure Single Sign-On* under *General Settings* in SAP Cloud for Customer administration view.
`Client Key`	Client ID Paste the entry you have copied from the *Client ID* field when configuring the OAuth client. For more information, see Configure the OAuth Client for OData Access [page 1236].
`Token Service URL`	**`https://<my_SAP_Cloud_for_Customer_system_name>.crm.ondemand.com/sap/bc/sec/oauth2/token`**

Parameter	Value
`Token Service User`	Client ID Paste the entry you have copied from the *Client ID* field when configuring the OAuth client. For more information, see Configure the OAuth Client for OData Access [page 1236].
`Token Service Password`	Client secret Paste the entry you have copied from the *Client Secret* field when configuring the OAuth client. For more information, see Configure the OAuth Client for OData Access [page 1236].

2. Configure the required additional properties. To do so, in the *Additional Properties* panel, choose *New Property*, and enter the following properties:

Parameter	Value
`authnContextClassRef`	**`urn:none`**
`nameIdFormat`	**`urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`**
`scope`	Scope ID entries separated by space. We recommend that you use **`UIWC:CC_HOME`**.
`userIdSource`	email

3. Select the *Use default JDK truststore* checkbox.

- To use basic authentication, create a *Basic Authentication* HTTP destination and configure its settings as follows:

Parameter	Value
`Name`	Enter a meaningful name
`Type`	**`HTTP`**
`Description`	(Optional) Enter a meaningful description
`URL`	**`https://<my_SAP_Cloud_for_Customer_system_name>.crm.ondemand.com/sap/c4c/odata/v1/c4codata`**
`Proxy Type`	**`Internet`**
`Authentication`	**`BasicAuthentication`**

Parameter	Value
`User`	Enter the name of the SAP Cloud for Customer user who should have access to the extension applications. This user will be used as a technical user.
`Password`	Enter the password of the SAP Cloud for Customer user who should have access to the extension applications.

4. Save your entries.

Results

It takes some time for the system to activate the destination.

Related Information

Create Destinations

5.4 Extending SAP Ariba in the Neo Environment

This document guides you through the configuration tasks that you need to perform to enable the use of the extension capabilities of SAP BTP for your SAP Ariba solution.

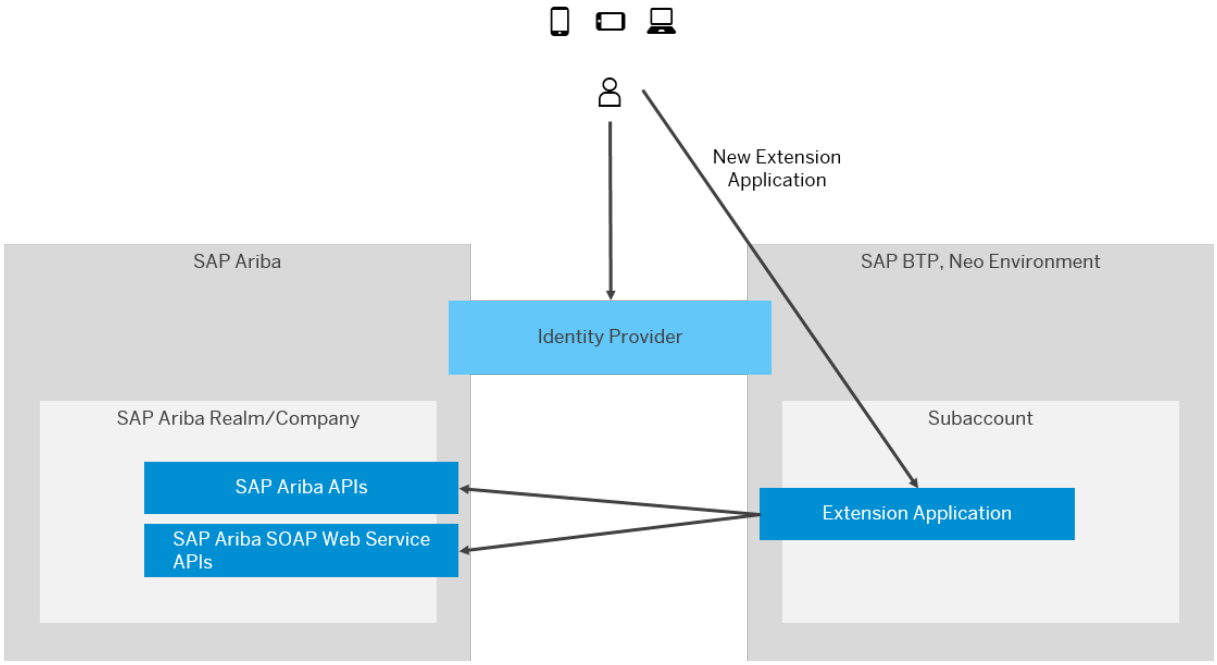
SAP BTP is the extension platform from SAP. It enables you to develop loosely coupled extension applications, thus implementing additional workflows or modules on top of the existing SAP Ariba solution you already have. Depending on the extension application and the SAP Ariba solution you use, you have an SAP Ariba Company/SAP Ariba Realm.

> **i Note**
>
> You can extend SAP Ariba only using the Neo environment of the SAP BTP.

The functionality is available for enterprise accounts.

In this scenario these are the important aspects:

- You need to set up the trust between SAP Ariba and an identity provider.
- You also need to set up the trust between SAP BTP and the same identity provider you used for the trust with SAP Ariba.
- The extension applications for SAP Ariba are hosted or subscribed in a dedicated subaccount in SAP BTP.



Procedure

1. Get a global account in the Neo environment. See Get an Enterprise Account.
2. Have an SAP Ariba solution.
   You can check the available SAP Ariba solutions at https://www.ariba.com/solutions/solutions-overview.
3. Configure Single Sign-On (SSO) [page 1241]:
   - Set Up Trust Between SAP Ariba and Corporate Identity Provider Using SAML 2.0 [page 1242].
   - Application Identity Provider.
4. Connect the Application Running on SAP BTP to SAP Ariba [page 1244].
   - Using SAP Ariba APIs [page 1244]
   - Using SAP Ariba SOAP Web Service APIs [page 1253]

5.4.1 Configure Single Sign-On (SSO)

Context

SAP BTP introduces Single Sign-On (SSO) through SAML2.0. In SAP BTP, identity information is provided by identity providers, and not stored on the cloud platform itself. If you have configured corporate identity provider

in SAP Ariba, you have to configure the same identity provider for SAP BTP to have seamless application login experience for the users. To enable this seamless login experience you need to:

- Perform a few configuration tasks on SAP BTP
- Change some of the SAP Ariba configurations if you don't use SAML 2.0 compliant authentication mechanism

Procedure

1. Configure SSO between SAP Ariba and your identity provider. See Set Up Trust Between SAP Ariba and Corporate Identity Provider Using SAML 2.0 [page 1242].
2. Configure SSO between SAP BTP and your identity provider. See Set Up Trust Between Corporate Identity Provider and SAP BTP [page 1242].

5.4.1.1 Set Up Trust Between SAP Ariba and Corporate Identity Provider Using SAML 2.0

Context

SAML 2.0 (Security Assertion Markup Language 2.0) is a version of the SAML standard for exchanging authentication and authorization data between security domains. You have to configure your SAP Ariba solution to use your corporate identity provider via SAML 2.0.

> **i** Note
>
> You must use SAML 2.0 since both SAP Ariba Solutions and SAP BTP support the SAML 2.0 standard. For further questions on configuring SSO using SAML 2.0, contact your SAP Ariba representative(s); you can also request from them a copy of the SAP Ariba Remote Authentication Deployment Guide.

5.4.1.2 Set Up Trust Between Corporate Identity Provider and SAP BTP

Use this procedure to configure the SAP BTP trust settings and to add your corporate identity provider.

Context

Procedure

1. Open the SAP BTP cockpit, select the region in which your subaccount is hosted, then select the global account that contains your subaccount, and then choose the tile of your subaccount. For more information about the regions, see Regions and Hosts.
2. Choose *Security* > *Trust* in the navigation.
3. Choose the *Local Service Provider* tab, and then choose *Edit*.
4. From the *Configuration Type* dropdown box, select *Custom*.
5. Configure the signing key and the signing certificate as follows:
   - If you want to use a signing key and a self-signed certificate automatically generated by the system, choose *Generate Key Pair*.
   - If you have your own key and certificate generated from an external application and signed by a trusted CA), you can use them instead of using the ones generated by the SAP BTP. To do so, copy the Base64-encoded signing key in the *Signing Key* field, and then copy the textual content of the certificate in the *Signing Certificate* field.
6. From the *Principal Propagation* dropdown box, select *Enabled*.
7. Choose *Save*.
8. Choose the *Get Metadata* link to download the SAP BTP metadata for your subaccount.
9. Save the metadata of your corporate identity provider on your local file system as an XML file. You will need this metadata in **Step 10**.
10. In the cockpit, choose *Security* > *Trust* > *Application Identity Provider*. Then choose *Add Trusted Identity Provider*. In the *General* tab, upload the SAP BTP metadata XML file (from **Step 9**) in the *Metadata File* field.
11. Choose *Save*.
12. Configure the SAML 2.0 trust with the subaccount on your corporate identity provider side. To do that, you have to upload the metadata XML file of your subaccount in SAP BTP that you have downloaded in **Step 8** in your corporate identity provider.

Results

The trust will be established automatically upon registration on both the SAP BTP and identity provider side.

Related Information

ID Federation with the Corporate Identity Provider

5.4.2 Connect the Application Running on SAP BTP to SAP Ariba

After you have configured the Single Sign-On (SSO), you need to configure the connectivity layer from SAP BTP to SAP Ariba. To do that, you have to use the SAP Connectivity service:

- It allows applications running on SAP BTP to access securely remote services that run on the cloud or on-premise.
- It also provides an API that application developers can use to consume remote services.
- It allows subaccount- or application-specific configuration of application connections via HTTP and Mail destinations.

For more information, see:

- Connectivity Service
- Consuming the Connectivity Service (Java)

You can configure connectivity to the following SAP Ariba APIs:

- Using SAP Ariba APIs [page 1244]
- Using SAP Ariba SOAP Web Service APIs [page 1253]

5.4.2.1 Using SAP Ariba APIs

Context

SAP Ariba APIs allow you to extend, integrate, and optimize SAP Ariba solutions to meet unique domain- or region-specific business requirements. By providing an open, secure, and scalable way to build new or extend existing functionality, SAP Ariba APIs deliver a significantly higher level of data sharing and functionality for your SAP Ariba procurement solutions. The expanding number of available APIs combined with the necessary tools and developer resources for rapid prototyping enables you to create custom solutions.

SAP Ariba APIs use different authentication mechanism depending on the API.

- OAuth 2.0 client credentials flow
- Basic authentication

> **i Note**
>
> To check which authentication mechanism uses the API you are interested in, go to SAP Ariba Developer Portal (USA region or Europe region).

Procedure

1. Get an SAP Ariba APIs Account [page 1245]
2. Configure SAP Ariba APIs [page 1246]
3. Configure the SAP Connectivity service to SAP Ariba APIs [page 1249]

5.4.2.1.1 Get an SAP Ariba APIs Account

Context

The SAP Ariba APIs account gives you access to a set of SAP Ariba APIs which enable you to build new applications and extend SAP Ariba functionality. With this account you get access to the SAP Ariba Developer Portal (USA region or Europe region) with a set of developer resources and tools.

Once you have an account on SAP Ariba APIs, you can:

- have access to different APIs documentation
- try out different APIs
- create new extension applications

Procedure

1. To get an SAP Ariba APIs account, go to the SAP Ariba Developer Portal (USA region or Europe region) and follow the instructions for sign up.
2. Explore available SAP Ariba APIs in the SAP Ariba Developer Portal (USA region or Europe region).

Next Steps

Configure SAP Ariba APIs [page 1246]

5.4.2.1.2 Configure SAP Ariba APIs

Context

To be able to consume the SAP Ariba APIs, you need to register a dedicated application in SAP Ariba Developer Portal (USA region or Europe region). For each registered application, an application key is generated. You can use it to try out an API and start developing your extension application running on SAP BTP. When developing this application, you need to work against sandbox environment with mocked API data. Once your extension application is ready, follow the instructions on https://developer.ariba.com/api/guides to enable the application registered in SAP Ariba Developer Portal for production access.

You can find detailed information about each API in the *Discovery* section at https://developer.ariba.com/api/apis (for the USA region) or at https://eu.developer.ariba.com/api/apis (for the Europe region).

Configure SAP Ariba APIs

You have an SAP Ariba APIs account and have explored the available APIs. You can start configuring the SAP Ariba APIs to be able to accept requests from your application.

First, you register an application running on SAP BTP in SAP Ariba Developer Portal (USA region or Europe region). Once you do so, you receive an application key. Using this key, you can start exploring the APIs requests and responses in SAP Ariba APIs sandbox environment.

To access real data from SAP Ariba systems, you request production access for your registered application. When production access is granted, you receive an OAuth client for OAuth authentication or service provider credentials for basic authentication (depending on the authentication mechanism a particular API uses) that has to be used when calling the API.

For a full list of the SAP Ariba APIs, see https://help.sap.com/viewer/p/ARIBA_APIS.

Procedure

1. Register an Application in SAP Ariba Developer Portal [page 1247].
2. Promote the Application in SAP Ariba APIs to Production [page 1248].

5.4.2.1.2.1 Register an Application in SAP Ariba Developer Portal

To be able to use SAP Ariba APIs, you have to register an application in SAP Ariba Developer Portal. This application has an API key. You use the API key with each call to an API published on SAP Ariba APIs sandbox or production environment.

Prerequisites

Have an SAP Ariba APIs account. See Get an SAP Ariba APIs Account [page 1245].

Context

Procedure

1. Log on to SAP Ariba Developer Portal (USA region or Europe region).
2. Choose *Create Application*.
3. Fill in the application name and its description and choose *Submit*.
4. Browse the *My Applications* list and find your application. Use the application key to start developing your application.

Next Steps

You can do one of the following:

- Test your application against the sandbox environment and configure the SAP Connectivity service to SAP Ariba APIs sandbox environment. See Using Sandbox Environment [page 1250].
- Make it productive and configure the SAP Connectivity service to SAP Ariba APIs production environment. See Promote the Application in SAP Ariba APIs to Production [page 1248] and Using Production Environment [page 1251].

5.4.2.1.2.2 Promote the Application in SAP Ariba APIs to Production

Prerequisites

- Have an SAP Ariba APIs account. See Get an SAP Ariba APIs Account [page 1245].
- Have your application registered in SAP Ariba Developer Portal (USA region or Europe region). See Register an Application in SAP Ariba Developer Portal [page 1247].

Context

You have registered an application running on SAP BTP in SAP Ariba Developer Portal and have explored different APIs published on SAP Ariba APIs sandbox environment. Then, you want to promote the registered application to production access.

A registered application promoted to production access in SAP Ariba Developer Portal is enabled for access to an SAP Ariba system and has an OAuth client, or a service provider account with credentials.

You use the OAuth client ID and client secret (in case of OAuth authentication) or the service provider credentials (in case of Basic authentication) to manage the authentication with an API published on SAP Ariba APIs production environment.

Procedure

1. Request production access for your registered application.
2. Find your application client ID. The client ID was generated when the production access was granted.
3. Generate a client secret. To execute an OAuth authentication, you will need the client secret for your application.

   > **i Note**
   >
   > Some APIs use Basic authentication and for them you don't need the client secret. You can find more information at SAP Ariba Developer Portal.

   For more details, follow the steps from SAP Ariba APIs guide:

   https://developer.ariba.com/api/guides (for the USA region) or https://eu.developer.ariba.com/api/guides (for the Europe region).

5.4.2.1.3 Configure the SAP Connectivity service to SAP Ariba APIs

Context

You can configure connectivity from SAP BTP to SAP Ariba APIs sandbox and production environment.

- APIs published on SAP Ariba APIs sandbox environment are meant for exploring and testing.
  To configure connectivity to an API published on SAP Ariba APIs sandbox environment, you create an HTTP destination for this API.
- APIs published on SAP Ariba APIs production environment are secured with OAuth or Basic authentication depending on the API .
  This means that your extension application may need connectivity to the SAP Ariba APIs OAuth Server as well.
  To configure connectivity to an API published on SAP Ariba APIs productive environment, you create:
  - an HTTP destination for this API
  - an HTTP destination for SAP Ariba APIs OAuth Server if needed

  > **i Note**
  >
  > We recommend that you create both HTTP destinations for the SAP Ariba APIs and for the SAP Ariba APIs OAuth Server on application level.

> **i Note**
>
> By updating the URL field in the HTTP destination for the API, you can easily switch from SAP Ariba APIs sandbox to production environment and vice-versa.

Procedure

1. Configure the SAP Connectivity service to SAP Ariba APIs Sandbox Environment. See Using Sandbox Environment [page 1250].
2. Configure the SAP Connectivity service to SAP Ariba APIs Productive Environment. See Using Production Environment [page 1251].

5.4.2.1.3.1 Using Sandbox Environment

Prerequisites

- You have an extension application in SAP Ariba APIs and a respective API key.
- You have explored the SAP Ariba APIs and are familiar with the API calls and the SAP Ariba APIs sandbox environment URL.

Context

You can develop and test your extension application running on SAP BTP against SAP Ariba APIs sandbox environment using this API key. The connectivity from the SAP BTP to an API published on SAP Ariba APIs sandbox environment is done via HTTP destination in SAP BTP.

For more information on how to develop a Java application running on SAP BTP, see Getting Started with Java Applications.

Procedure

1. Create an HTTP destination for an SAP Ariba API in SAP BTP cockpit on application level. See Create HTTP Destinations.
2. Use the following values for the HTTP destination:

Parameter	Value
Name	Enter a meaningful name.
Type	HTTP
Description	(Optional) Enter a meaningful description.
URL	`https://openapi.ariba.com/api/<service_name>/<version>/sandbox`
Proxy Type	Internet
Authentication	NoAuthentication
apiKey	Enter the SAP Ariba APIs registered application API key.

5.4.2.1.3.2 Using Production Environment

Prerequisites

- Have developed and tested your extension application against SAP Ariba APIs sandbox environment.
- Have an application registered in SAP Ariba Developer Portal (USA region or Europe region) promoted for production access.

Context

Until now you have developed and tested your extension application against SAP Ariba APIs sandbox environment.

Now you want your application to work against your productive SAP Ariba system. You have promoted your SAP Ariba APIs application for production access. You need to configure the SAP Connectivity service to SAP Ariba APIs productive environment.

Procedure

1. If the API uses OAuth, create an HTTP destination to the SAP Ariba APIs OAuth Server in SAP BTP cockpit on application level. Use the following values:

Parameter	Value
Name	Enter a meaningful name.
Type	HTTP
Description	(Optional) Enter a meaningful description.
URL	`https://api.ariba.com/v2/oauth/token`
Proxy Type	Internet
Authentication	BasicAuthentication
User	Enter the SAP Ariba APIs registered application OAuth client ID.
Password	Enter the SAP Ariba APIs registered application OAuth client secret.

2. Create another HTTP destination to an SAP Ariba APIs in SAP BTP cockpit on application level or update the destination that you have used for the sandbox environment. Use the following values:

Parameter	Value
Name	Enter a meaningful name.
Type	HTTP
Description	(Optional) Enter a meaningful description.
URL	`https://openapi.ariba.com/api/<service_name>/<version>/prod`
Proxy Type	Internet
Authentication	Depending on the API you need one of these options: ○ NoAuthentication, if the API requires OAuth authentication. ○ BasicAuthentication, if the API requires Basic authentication. Use the same credentials as in the service provider.
apiKey	Enter the registered application API key.

3. If the API uses OAuth authention, update your extension application to use OAuth 2.0 authentication.

   To access protected resources with OAuth 2.0, you need to acquire an access token. Find more information at https://developer.ariba.com/api/guides (for the USA region) or https://eu.developer.ariba.com/api/guides (for the Europe region), in the **Authentication** section.

5.4.2.1.4 Sample SAP Ariba APIs Extension Applications

Prerequisites

Have configured connectivity from SAP BTP to SAP Ariba APIs.

Context

You can use these sample extension applications as a reference when developing your extension application running on SAP BTP that consumes the SAP Ariba APIs.

Sample	Description
SAP Ariba Discovery RFx to External Marketplace Extension https://github.com/SAP/cloud-ariba-discovery-rfx-to-external-marketplace-ext	SAP Ariba Public Sourcing is a sample extension application for SAP Ariba Network that runs on SAP BTP. The purpose of the application is to collect public sourcing events from SAP Ariba Discovery via SAP Ariba APIs and to display them in an application running on SAP BTP. Extends: SAP Ariba Network Uses: Discovery RFx Publication to External Marketplace API
SAP Ariba Partner Flow Extension https://github.com/SAP/cloud-ariba-partner-flow-extension-ext	SAP Ariba Partner Flow extension application allows a buyer to attach documents to Advance Ship Notices (ASNs). Extends: SAP Ariba Network Uses: Partner Flow Extension API

5.4.2.2 Using SAP Ariba SOAP Web Service APIs

Context

SAP Ariba SOAP Web Service APIs enable you to exchange data between SAP Ariba solutions and other systems for real-time data integration. You can use the SOAP Web Service APIs to extend and integrate SAP Ariba solution to meet your unique domain specific needs.

The following instructions guide you on how to configure connectivity from SAP BTP to an SAP Ariba solution with enabled SOAP Web Service APIs.

> **i Note**
>
> You can check the Sample Web Services Extension Applications [page 1257] for more concrete examples.

Procedure

1. Configure the SAP Ariba Solution [page 1254].
2. Connect SAP BTP to SAP Ariba SOAP Web Service APIs [page 1256].

5.4.2.2.1 Configure the SAP Ariba Solution

Prerequisites

- Have an SAP Ariba solution with enabled SOAP Web Service APIs.
  To get more information on the available SOAP Web Service APIs, contact your SAP Ariba representative.
- Be a member of the Customer Administrator or Integration Admin group, or a group with the Administrator or Integration Admin role in the SAP Ariba solution.

Context

You configure your SAP Ariba solution by creating new integration end point and enabling an integration task.

5.4.2.2.1.1 Create New Integration End-Point

Context

You create an integration end-point in your SAP Ariba solution.

Procedure

1. Login into your SAP Ariba solution.
2. Click the expansion arrow for *Integration Manager*.

   > **i** Note
   >
   > For example, in the Procure-to-Pay solution, you have to choose *Manage* > *Core Administration* > *Integration Manager*.

3. Choose *End Point Configuration*.
4. Choose *Create New*. An *End Point Configuration - Create End Point* page opens.
5. In the *Name* field, enter a name for the end-point.
6. In the *Type* filed, choose *Inbound*.
7. In the *HTTP Authentication* section, enter a user ID in the *Login* field and a password in the *Password* field.
   You will use the user ID and password to authenticate your HTTP calls later.

8. Save the changes.

Results

You have created an integration end-point. Now you have to enable an integration task and link it to the integration end-point.

5.4.2.2.1.2 Enable an Integration Task

Context

You enable an integration task and link it to an already created end-point.

Procedure

1. Login into your SAP Ariba solution.
2. Click the expansion arrow for *Integration Manager*.

   > **i Note**
   >
   > For example, in the Procure-to-Pay solution, you have to choose *Manage* > *Core Administration* > *Integration Manager*.

3. Choose *Integration Configuration*.
4. Find the task you want to enable. Select *List All* or enter search criteria and click *Search*.
5. Choose *Actions* > *Edit* for this particular task. An *Edit data import/export task* page opens.
6. For the *Status* field, select *Enabled*.
7. Use the drop-down menu for the *End point* field to select the end- point you have already created.

   > **i Note**
   >
   > - You can view the WSDL file from the *View WSDL* link.
   > - Pay attention on the Integration Task *URL* field. This is the web service end-point.

8. Save the changes.

Results

You have successfully enabled an integration task and have linked it to an integration end-point. You can now call the SAP Ariba SOAP Web Service API related to the integration task.

Related Information

SAP Ariba Strategic Sourcing solutions
Common data import and administration guide for SAP Ariba Strategic Sourcing and Supplier Management solutions
SAP Ariba Procurement solutions

5.4.2.2.2 Connect SAP BTP to SAP Ariba SOAP Web Service APIs

Prerequisites

- Have a new integration end-point created. See Create New Integration End-Point [page 1254].
- Have an integration task enabled. See Enable an Integration Task [page 1255].

Context

You configure connectivity in SAP BTP by creating an HTTP destination.

Procedure

1. Create an HTTP destination in SAP BTP cockpit on application level. See Create HTTP Destinations.
2. Use the following values for the HTTP destination:

Parameter	Value
Name	Enter a meaningful name.

Parameter	Value
Type	HTTP
Description	(Optional) Enter a meaningful description.
URL	Enter the integration task URL.
Proxy Type	Internet
Authentication	BasicAuthentication
User	Enter the integration end point user ID.
Password	Enter the integration end point password.

> **i** Note
>
> Some of the SOAP Web Service APIs support Basic authentication. Others support authentication without Base64 encoding. Example Authorization headers:
>
> - Authorization: Basic base64 (userID:password)
> - Authorization: Basic userID:password
>
> For more information on the supported authentications, see:
>
> - SAP Ariba Strategic Sourcing solutions
> - SAP Ariba Procurement solutions

5.4.2.2.3 Sample Web Services Extension Applications

Prerequisites

Have configured connectivity from SAP BTP to SAP Ariba SOAP Web Service API.

Context

You can use these sample extension applications as a reference when developing your extension application running on the SAP BTP that consumes the SAP Ariba SOAP Web Service API.

Sample	Description
Requisition Client for the SAP Ariba Procure-to-Pay Solution https://github.com/SAP/cloud-ariba-p2p-requisition-client-ext	You can use this sample to submit requisitions to your SAP Ariba instance. Extends: SAP Ariba Procure-To-Pay Uses: Import Requisition(s) API
Catalog Client for the SAP Ariba Procure-to-Pay Solution https://github.com/SAP/cloud-ariba-p2p-catalog-client-ext	You can use this sample to make a catalog search for a specified item in your SAP Ariba instance. Extends: SAP Ariba Procure-To-Pay Uses: Catalog Item Search API
SAP Ariba Simple Requisition Extension https://github.com/SAP/cloud-ariba-simple-requisition-ext	This is a Web application running on SAP BTP that demonstrates the consumption of SAP Ariba Procure-to-Pay API for purchasing requisitions. Extends: SAP Ariba Procure-To-Pay Uses: Import Requisition(s) API
SAP Ariba QR Code Requisition Extension https://github.com/SAP/cloud-ariba-qrcode-requisition-ext	This is a Web application running on SAP BTP that shows how to search for catalog items and submit requisitions using the SAP Ariba Procure-to-Pay API. There are two possible usage scenarios for this application: • Scan a QR Code to show details about a specified item and submit a requisition through the application UI. • Scan a QR Code to submit requisition without starting the application UI. Extends: SAP Ariba Procure-To-Pay Uses: Catalog Item Search API, Import Requisition(s) API

6 Administration and Operations, Neo Environment

Learn about the different account administration and application operation tasks which you can perform in the Neo environment.

- Account Administration in the Cockpit [page 1259]
- Account Administration Using the SAP BTP Command Line Interface (btp CLI) [Feature Set B] [page 1317]
- Account Administration in the Neo Console Client [page 1346]
- Application Operations in the Neo Environment [page 1587]

6.1 Account Administration in the Cockpit

Learn about frequent administrative tasks you can perform using the SAP BTP cockpit, including managing subaccounts, entitlements and members.

- Managing Global Accounts Using the Cockpit [page 1261]
- Managing Entitlements and Quotas Using the Cockpit [page 1291]

Related Information

Account Administration in the Neo Console Client [page 1346]

6.1.1 Navigate in the Cockpit

Learn how to navigate to your global accounts and subaccounts in the SAP BTP cockpit.

Context

Procedure

1. Find out which cloud management tools feature set your global account uses. For more information, see Cloud Management Tools — Feature Set Overview.
2. Become familiar with the navigation behavior in the SAP BTP cockpit.
   - Navigate in the Cockpit to Global Accounts and Subaccounts [Feature Set A] [page 1260]
   - Navigate in the Cockpit to Global Accounts and Subaccounts [Feature Set B] [page 1260]

Navigate in the Cockpit to Global Accounts and Subaccounts [Feature Set A]

Procedure

1. Navigate to a global account.

   You can see the following path in the breadcrumbs:

   Home / <global_account>
2. Navigate to a subaccount.
   a. Select the global account that contains the subaccount you'd like to navigate to by following the steps described above.
   b. Select the subaccount.

   You can see the following path in the breadcrumbs:

   Home / <global_account> / <subaccount>

Navigate to Global Accounts, Directories, and Subaccounts [Feature Set B]

Procedure

1. Navigate to a global account.
   - **You have only one global account:**
     When you log on to the cockpit you're automatically taken into your global account, to the *Account Explorer* page.
   - **You have multiple global accounts:**
     When you log on to the cockpit, a dialog opens up containing the list of all global accounts that you're part of. In this dialog, select the global account you want to navigate to.

> → Tip
>
> If you have one global account you usually navigate to, you can choose *Remember my selection* in the initial dialog. This means you'll be automatically redirected to your preferred global account after logging on, without seeing the dialog with all the options each time.
>
> If you've chosen a default global account, you can change or remove it anytime. To do so, navigate to the *Account Explorer* page of any global account, choose *Switch Global Account*.
>
> To change it, choose the desired new default global account from the list, select *Save new selection as default global account* and choose *Continue*. Your new default is savedand you're redirected to that global account.
>
> To delete the default global account and go back to seeing the selection dialog after each logon, simply choose the ⊗ icon next to your default global account name in the dialog and choose *Close*.

You can see which global account you are in at any time by looking at the first item in the breadcrumbs. It looks like this: 🌐 *<global account name>*

2. Navigate to a different global account.
   - Navigate to the *Account Explorer* page at global account level and choose *Switch Global Account*.
   - Use the dropdown menu next to the 🌐 *<global account name>*.
3. (Optional) Navigate to directories.
   a. When you enter your global account, you are by default taken to the *Account Explorer* page of that global account. To navigate to a directory, choose the corresponding entry in one of the views in the *Directories & Subaccounts* tab.
4. Navigate to subaccounts.
   a. When you enter your global account, you are by default taken to the *Account Explorer* page of that global account. To navigate to a subaccount, choose the corresponding entry from one of the views.

      Once you've entered a subaccount, the breadcrumbs look like this: 🌐 *<global account name>* / 🖧 *<subaccount name>*

6.1.2 Managing Global Accounts Using the Cockpit

Your SAP BTP global account is the entry point for managing the resources, landscape, and entitlements for your departments and projects in a self-service manner.

Feature Set B: Global Account Settings

As a global account administrator, you can access the global account settings by clicking ⚙ in the SAP BTP cockpit.

In the *General* tab of the global account settings, you can identify your global account subdomain.

In the *Subaccount Defaults* tab in the global account settings, you can set the supported providers, the default provider, and the default region. These defaults are used when creating a new subaccount in this global account.

Related Information

Setting Up Your Account Model
Managing Directories Using the Cockpit [Feature Set B] [page 1277]
Managing Subaccounts Using the Cockpit [page 1282]
Using Subaccounts to Create a Staged Development Environment

6.1.2.1 Log On to Your Global Account

Use the SAP BTP cockpit to log on to your global account and start working in SAP BTP.

Prerequisites

You have received a welcome e-mail from SAP for your global account.

Context

When your organization signs a contract for SAP BTP services, an e-mail is sent to the e-mail address specified in the contract. The e-mail message contains the link to log on to the system and the SAP Cloud Identity credentials (user ID) for the specified user. These credentials can also be used for sites such as the SAP Store or the SAP Community.

Procedure

1. Use the link in your Welcome e-mail.

   When using cloud management tools feature set A: Alternatively, for example, choose https://account.eu1.hana.ondemand.com.

   When using cloud management tools feature set B: Alternatively, for example, choose https://cockpit.eu10.hana.ondemand.com/cockpit/.

   To avoid latency, make sure you choose a logon URL in the region closest to you.

> **Note**
>
> If single sign-on has not been configured for you, you will have to enter your credentials. You'll find your logon ID in your Welcome e-mail.

2. Choose a global account.

Results

When using cloud management tools feature set A: The global account *Overview* page opens.

When using cloud management tools feature set B: The *Account Explorer* page for that global account opens.

Next Steps

- Track and monitor usage of services in your global account:
  - Use the filters to specify which usage information to display in the tables and monthly usage charts. The *Period* filter is applied to the chart display.
  - Some rounding or shortening is applied to large values. Mouse over values in the table to view the exact values in the tooltips.
  - Choose a row in the table to view its historic information in the *Monthly Usage* chart. To display a larger view of a chart, choose the *(Zoom)* button.
  - To download the information in the *Overview* page to a Microsoft Excel file, choose the *Export* button.
- Access your subaccounts and create new subaccounts.
  See Create a Subaccount.

Related Information

Cloud Management Tools — Feature Set Overview
Account Model
Global Accounts
Create a Subaccount

6.1.2.2 Change the Display Name of Your Global Account

Change the display name for the global account using the SAP BTP cockpit.

Context

> → Tip
>
> To find out how to tell which cloud management tools feature set your global account uses, see Cloud Management Tools — Feature Set Overview.

Changing the global account name

Prerequisites

You are a member of the global account that you'd like to edit.

Context

The overview of global accounts available to you is your starting point for viewing and changing global account details in the cockpit. To view or change the display name for a global account, trigger the intended action directly from the tile for the relevant global account.

Procedure

1. Choose the global account for which you'd like to change the display name and choose  on its tile.

   A new dialog shows up with the mandatory *Display Name* field that is to be changed.
2. Enter the new human-readable name for the global account.
3. Save your changes.

Changing the global account name

Prerequisites

You are an admin of the global account that you'd like to edit.

Context

> → Tip
>
> You can also change the account's display name using the SAP BTP command line interface (btp CLI) with the `btp update accounts/global-account` command. See Account Administration Using the SAP BTP Command Line Interface (btp CLI) [Feature Set B] [page 1317].

Procedure

1. Open your global account in the SAP BTP cockpit.
2. Navigate to the *Account Explorer* page.
3. In the *Directories & Subaccounts* view, locate the first entry in your account structure, which is your global account, and click ••• *Actions*. Then select *Edit*.
4. In the *Edit Global Account* dialog box, enter the new human-readable name for the global account.
5. Save your changes.

Related Information

Account Model [page 10]

6.1.2.3 Managing Resource Providers

SAP BTP allows you to connect your global account in the SAP BTP cockpit to your provider account from a non-SAP cloud vendor, and consume remote service resources that you already own and which are supported by SAP through this channel.

> ℹ Note
>
> The use of this functionality is subject to the availability of the supported non-SAP cloud vendors in your country or region.

Context

For example, if you are subscribed to Amazon Web Services (AWS) and have already purchased services, such as PostgreSQL, you can register the vendor as a resource provider in SAP BTP and consume this service across your subaccounts together with other services offered by SAP.

SAP BTP currently supports the following vendors and their consumable services:

Cloud Vendor	Supported Services
Amazon Web Services	Amazon Relational Database Service (RDS) - PostgreSQL
Microsoft Azure	Azure Database for PostgreSQL

More information: PostgreSQL on SAP BTP - Product Documentation

Configure a New Resource Provider

Context

To consume the resources provisioned by your provider account, you need to first create a resource provider instance in the cockpit.

Procedure

1. Navigate to your global account in the cockpit.
2. Choose *Resource Providers* from the left hand-side navigation panel.
3. Choose *New Provider*.
4. Choose the provider from the list of supported vendors.
5. Enter a display name for the provider.

   You can create more than one instance of a given resource provider, each with its unique configuration properties. In such cases, the display name (and technical name) should be descriptive enough so that you and developers can easily differentiate between each instance.
6. Enter a unique technical name for the provider.

   The technical name can contain only letters (a-z), digits (0-9), and underscore (_) characters.

   This name can be used by developers as a parameter when creating service instances from this provider.

   > **i Note**
   >
   > After you save the settings for the resource provider, you cannot change its technical name.
7. Choose either *Manually enter provider configuration properties* or *Add provider configuration properties by JSON file*, depending on how you prefer to provide the necessary configuration properties for the given provider.

> → Tip
>
> Each supported vendor has its own unique configuration properties. The *New Resource Provider* form provides a description of each property and indicates whether they are mandatory or optional.

If you choose to configure the properties by uploading a JSON file, which is typically provided by the vendor, make sure that the file contains the required configuration properties for connecting to the provider, and is correctly formatted.

> Sample Code
>
> Sample JSON for configuring Amazon Web Services (AWS) as a resource provider:
>
> ```
> {
> "access_key_id": "AWSACCESSKEY",
> "secret_access_key": "SECRETKEY",
> "vpc_id": "vpc-test",
> "region": "eu-central-1"
> }
> ```

8. Save your changes.

   The new resource provider is added to the resource providers table.

Results

After you configure a new resource provider, its supported services are added as entitlements in your global account. In the *Entitlements* page in the cockpit, you can then allocate the required services and quotas to the relevant directories and subaccounts in your global account.

Manage Resource Providers and Service Entitlements

Context

Follow these steps to manage the resource providers that you have already configured in the cockpit.

Procedure

1. Navigate to your global account in the cockpit.
2. Open the *Resource Providers* page.

   The page displays the resources providers that you have already configured.
3. In the *Actions* columns, you can perform the following for each resource provider:

Action	Description
Manage Entitlements and Quotas	View the subaccount entitlements and quotas of services that are consumed from a resource provider. This action opens the *Entitlements* > *Service Assignments* page in the cockpit, and is prefiltered for the selected resource provider. → Tip Whenever you need to manage assigned entitlements, you can navigate directly to the *Service Assignments* or *Subaccount Assignments* pages in the cockpit without going through the *Resource Providers* page.
Edit Resource Provider	Edit the display name, description, and configuration properties of a resource provider. i Note You cannot change the technical name of a resource provider.
Delete	Delete a resource provider. When you delete an existing resource provider, all the services offered by this instance of the resource provider will no longer be available on SAP BTP. i Note You cannot delete a resource provider that already has service entitlements assigned to subaccounts in your global account. To delete the provider, you need to first remove its subaccount assignments in the *Subaccount Assignments* page.

Related Information

Configure Entitlements and Quotas for Subaccounts

6.1.2.4 Monitoring Usage and Consumption Costs in Your Global Account

In a global account that uses the consumption-based commercial model, you can monitor the usage of billed services and your consumption costs in the SAP BTP cockpit.

i Note

- The use of the consumption-based commercial model is subject to its availability in your country or region.
- If your global account uses the subscription-based commercial model, then information in this topic that refers to usage data also applies to you. Ignore information about costs and cloud credits.

- Directories only apply to global accounts that are running in feature set B.

Accessing Usage Analytics for Your Global Account

Accessing the global account's usage analytics is different when using cloud management tools feature set A or B. For more information on the two feature sets, see Cloud Management Tools — Feature Set Overview.

Feature Set A

To monitor and track usage in your global account, open the global account in the cockpit and choose *Overview* in the navigation area.

The global account's *Overview* page provides information about the usage of cloud credits (if your account has a cloud-credit balance) and costs for a global account, and the usage of services in the global account and its subaccounts.

Feature Set B

To monitor and track usage in your global account, open the global account in the cockpit and choose *Usage Analytics* in the navigation area.

The global account's *Usage Analytics* page provides information about the usage of cloud credits (if your account has a cloud credits balance) and costs for a global account, and the usage of services in the global account and its directories and subaccounts.

Using the Global Account's Usage Analytics / Overview Page

i Note

- Global accounts that use the consumption-based commercial model can include actual usage information on the Cloud Foundry space level, which is different from billed usage used in your billing document. Usage data is processed according to accounting formulas that generate a billing document that aggregates all usage, from all spaces, so that it is favorable to customers. There is no unified method to calculate costs based on actual usage.
- Costs are displayed according to your contract currency.
- If your global account uses the subscription-based commercial model exclusively, then only usage data is displayed. Cloud credits and cost information is not relevant for such accounts.
- If your global account is licensed for both the consumption-based and subscription-based commercial models, the *Global Account Costs*, *Cloud Credits Usage*, and *Global Account Overview* views in this page show billing and usage data that is charged solely according to the consumption-based commercial model. For services in your subscription-based commercial agreement, you'll see billing and usage data for usage that exceeds your subscribed quota. For any excess service usage, you are charged according to your contract for the consumption-based commercial model.
  For example, if your subscription contract is entitled to consume a given service at a fixed cost for up to 100 unique site visits, and 151 unique site visits are registered, these views will show data relating only to the 51 visits that have exceeded the 100 limit.

When viewing usage data on the service or subaccount level for such accounts in the remaining views in this page, the displayed data is the total combined usage for both consumption-based and subscription-based commercial models.

- When there is a cloud-credit balance for the global account, the cloud-credit usage information is displayed for the current contract period. The total contract duration is split into contract periods (usually of one year each) and the total cloud credits are divided between these periods.

  > **i Note**
  >
  > If your global account has received a cloud-credit refund at any time during the current contract phase, you may see a difference between your total usage and monthly usage in the chart.

- When there is no cloud-credit balance for the global account, the monthly total costs for the global account are displayed from the contract start date.

  > **i Note**
  >
  > If your global account has received a refund at any time during the current contract phase, you may see a difference between the displayed monthly costs and your billing document.

- In the resource usage views, use the filters to specify which information to display. The *Period* filter applies only to the chart display.
- Some rounding or shortening is applied to large values. Mouse over values in the table to view the exact values in the tooltips.
- Choose a row in the table to view its historic information in the *Monthly Usage* chart.
- To display a larger view of a chart, choose the *(Zoom)* button.

Understanding the Views

View	Description	Action
Global Account Info	Displays general information about the global account, including the number of directories and subaccounts it contains, and the number of regions which have these subaccounts.	[Feature Set B]: Click the *Directories* or *Subaccounts* links to navigate directly to the *Account Explorer* page. [Feature Set A]: Click the *Subaccounts* link to navigate directly to the *Subaccounts* page.

View	Description	Action
Cloud Credits Usage (Only displayed when there is a cloud-credit balance for a global account that uses the consumption-based commercial mode.)	Displays the current balance and monthly usage of cloud credits as a percentage of the cloud credits allocated per contract period. • Cloud-credit information is based on the monthly balance statements for the global account. • For the time period between the last balance statement and the current date, the chart uses estimated values, which are displayed as striped bars. These estimates are based on resource usage values before computation for billing, and might change after the next balance statement is issued. The estimated values are not projected or forecast values. Alerts are displayed when you have used more than 90% and when you reach 100% of your cloud credits.	
Global Account Costs (Only displayed when there is no cloud-credit balance for a global account that uses the consumption-based commercial mode.)	Displays the total cost per month for usage of services in the global account from the contract start date. All cost information is for all regions in the global account. Actual costs are updated after the monthly statement is generated.	

View	Description	Action
Global Account Overview	Displays usage and costs of all services in the global account. The information is broken down according to service plans. All the regions in which a service plan is available are displayed; however, actual usage is only in the regions of your subaccounts. **i Note** Usage and cost information is updated every 24 hours. Data for the first of the month might not appear until the following day. The charts display information for each month for all service plans in the table or only for the selected row. Provisional data for the current month is displayed as striped bars. **i Note** Some service plans have differential pricing depending on the region. Usage and costs of a service plan in multiple regions with differential pricing are displayed as separate items.	Use the *View By* options to switch the view between service plan usage and costs for the global account. Use the filter to select the environment and service for which you want to view information. For the chart display, select a row, and filter by period. If no row is selected, the chart displays information for all the service plans in the table.
Service Usage	Displays resource usage according to service. Some service plans may display usage according to multiple metrics.	Use the filter to select the service and directory or subaccount for which to display usage. For the chart display, select a row, and filter by period. If no row is selected, the chart displays information for all the service plans in the table.
Directory/Subaccount Usage	Displays resource usage according to subaccount or directory. Some service plans may display usage according to multiple metrics.	Use the filter to select the directory or subaccount for which to display usage. For the chart display, select a row and filter by period. If no row is selected, the chart displays information for all the service plans in the table.

Exporting Usage and Cost Data Information

You can export the displayed usage and cost data to a spreadsheet document:

- To export usage and cost data from all the views, choose *Export* > *All Data*.
- To export only cloud-credit usage data, choose *Export* > *Cloud Credits Usage*.

The document with the relevant data is downloaded. The document contains several sheets (tabs). The sheets that are included in the export depend on the commercial model that is used by your global account (and the export option that you selected).

		Commercial Model	
Sheet Name	**Description**	**Subscription-Based**	**Consumption-Based**
Global Account Info	Provides general information about your global account. If there is a cloud-credit balance for the global account, then its cloud-credit usage, per month as a percentage of your total cloud credits for the current contract period, is also shown.	Yes	Yes
Global Account Costs	Allows you to view total monthly usage data and costs for all billable services and plans consumed at the level of your global account. The items listed are all the billed items that are created in the accounting system and deducted from the cloud-credit balance of your global account.	No	Yes
Subaccount Costs by Service	Allows you to view monthly usage data and costs of all billable services consumed by plan and subaccount. All subaccount calculations are estimated and proportionate to the total global account usage: `[Subaccount usage / Global account usage x Rate plan per SKU]` **i Note** Global accounts are the only contractual billable entity for SAP BTP. Directories and subaccounts are used as structural entities in global account, and their usage and cost data should only be used for your internal cost estimations. The relative calculation per billable usage within each subaccount is an estimation only as it is based on certain measures which in some cases can either be different from the metrics that are presented in the *Global Account Costs* tab or which use different formulas than the ones used for billing. This sheet is not included in the spreadsheet if your global account is licensed for both the consumption-based and subscription-based commercial models.	No	Yes

Sheet Name	Description	Commercial Model	
		Subscription-Based	Consumption-Based
Actual Usage	Allows to you to view the actual monthly usage data of all consumed services by plan and subaccount. Actual or raw usage data is different to the billed usage that is used in your billing document, and includes non-billable services. The aggregated usage for each service is based on a formula that is specific to each service, for example, MIN, MAX, or AVG.	Yes	Yes

Example

A global account, which uses the consumption-based commercial model, has the following usage data in a given month, where SAP HANA and Application Runtime are billable services, and Destination is a non-billable service:

Subaccount 1	1x SAP HANA 256 GB, 200 MB of Application Runtime, and 300 API calls of Destination
	200 MB of Application Runtime and 300 API calls of Destination
Subaccount 2	1x SAP HANA 512 GB, 600 MB of Application Runtime, and 300 API calls of Destination

Based on these usage measurements, you would see the following data in the spreadsheet. Each listed item represents one row in the spreadsheet. Cost prices shown are for illustration purposes only and do not reflect the actual rates for the mentioned services.

Sheet	Sample Usage and Cost Data Displayed in Spreadsheet
Global Account Costs	• 1 instance of SAP HANA 256 = EUR 1024 • 1 instance of SAP HANA 512 = EUR 2048 • 1 GB of Application Runtime = EUR 100 + estimated cost distribution to subaccounts
Subaccount Costs by Service	• Subaccount 1: 1 instance of SAP HANA 256 = EUR 1024 • Subaccount 2: 1 instance of SAP HANA 512 = EUR 2048 • Subaccount 1: 400 MB of Application Runtime = EUR 40 (400 MB / 1 GB x EUR 100) • Subaccount 2: 600 MB of Application Runtime = EUR 60 (600 MB / 1 GB x EUR 100)
Actual Usage	• Subaccount 1: 1 instance of SAP HANA 256 • Subaccount 2: 1 instance of SAP HANA 512 • Subaccount 1: 400 MB of Application Runtime • Subaccount 2: 600 MB of Application Runtime • Subaccount 1: 600 API calls of Destination • Subaccount 2: 300 API calls of Destination

Related Information

Cloud Management Tools — Feature Set Overview
What Is the Consumption-Based Commercial Model?
View Subaccount Usage Analytics [page 1288]
View Directory Usage Analytics [Feature Set B] [page 1281]
Monitoring Usage Information Using APIs of the SAP Usage Data Management Service [Feature Set B]

6.1.2.5 Configure Legal Information

Administrators can define legal links per enterprise global account in the SAP BTP cockpit.

Prerequisites

You have the Administrator role for the global account for which you'd like to define legal links.

Context

You can define the legal information relevant for a global account so the members of this global account can view this information.

You can define the privacy link relevant for a global account so the members of this global account can view this information.

Procedure

1. Choose the global account for which you'd like to make changes.
2. In the left-hand navigation, choose *Legal Information* > *Configure Legal Links*.
3. For each category, enter the relevant URL.
4. Save your changes.

   The links you configured are available in the *Legal Information* menu.

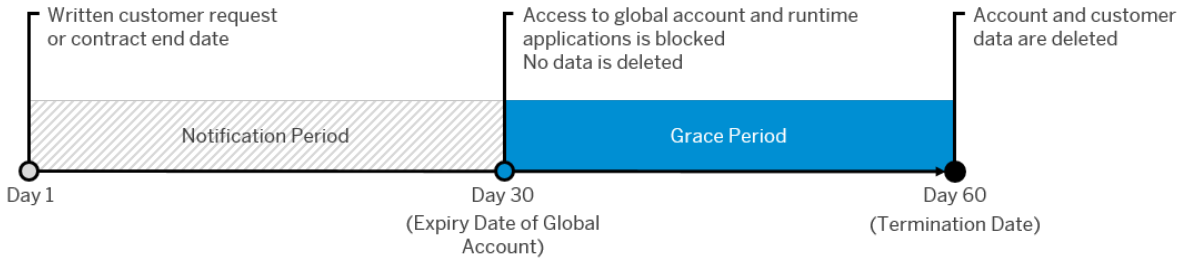
Related Information

Global Accounts [page 10]

6.1.2.6 Global Account Termination

When the contract of an SAP BTP customer ends, SAP is legally obligated to delete all the data in the customer's accounts. The data is physically and irreversibly deleted, so that it cannot be restored or recovered by reuse of resources.

The termination process is triggered when a customer contract expires or a customer notifies SAP that they wish to terminate their contract.



1. When the termination process is triggered, SAP sends a notification e-mail to the customer regarding the expired contract, and the dates on which the account will be blocked and then deleted.
2. During the 30-day notification period before the global account is closed, a banner is displayed in the SAP BTP cockpit reminding users of the global account about the pending termination.
   The customer can export their data, or they can open an incident to download their data before the termination date.
   The customer can cancel the termination during this period and renew their contract with SAP.
3. After the notification period, a grace period of 30 days begins.
   During the grace period:
   - Access is blocked to the global account, and to deployed and subscribed applications.
   - No data is deleted, and backups are still ongoing.
   - The global account is displayed in the cockpit with the label *Expired*, and the number of days left in the grace period is displayed.

   The customer can contact SAP to restore their account to a fully active account without data loss.
4. At the end of the grace period, all customer-related data for the account and services is deleted and cannot be restored. The global account tile is removed from the cockpit.

Related Information

Getting Support, Neo Environment [page 1859]

6.1.3 Managing Directories Using the Cockpit [Feature Set B]

Learn how to organize and manage your subaccounts according to your technical and business needs by using directories in the SAP BTP cockpit.

Context

> **i** Note
>
> This feature is new in Feature Set B so there is no equivalent in Feature Set A. For more information, see Cloud Management Tools — Feature Set Overview.

Procedure

1. Learn how to Create a Directory [Feature Set B] [page 1277].
2. Learn how to Manage the Account Explorer Hierarchy [Feature Set B] [page 1279].
3. Learn how to Manage Users in Directories [Feature Set B] [page 1280].
4. Learn how to View Directory Usage Analytics [Feature Set B] [page 1281].

Related Information

Directories [Feature Set B] [page 12]
Managing Subaccounts Using the Cockpit [page 1282]

6.1.3.1 Create a Directory [Feature Set B]

Create a directory using the SAP BTP cockpit to organize and manage your subaccounts. For example, you can group subaccounts by project, team, or department.

Context

> **i** Note
>
> This feature is new in Feature Set B so there is no equivalent in Feature Set A. For more information, see Cloud Management Tools — Feature Set Overview.

For more information on directories, see Directories [Feature Set B] [page 12].

Procedure

1. In your global account, navigate to the *Account Explorer* page.
2. In the *Create* dropdown, choose *Directory*.
3. In the dialog, enter a display name for your new directory and choose a parent.
4. (Optional - Set Custom Properties) Assign custom properties to the directory to make organizing and filtering your directories easier.
5. Review the details of your directory and choose *Create* to finalize the process.

Results

Your directory is created.

Next Steps

Once you've created a directory you can perform the following actions:

- Edit the directory - you can edit the name, description, entitlements and custom properties of the directory.
- Add subaccounts to the directory - Manage the Account Explorer Hierarchy [Feature Set B] [page 1279].
- Delete the directory - you can only delete a directory that contains no subaccounts.

Related Information

Cloud Management Tools — Feature Set Overview
Directories [Feature Set B] [page 12]
Manage the Account Explorer Hierarchy [Feature Set B] [page 1279]
Manage Users in Directories [Feature Set B] [page 1280]
Configure Entitlements and Quotas for Directories [Feature Set B]
View Directory Usage Analytics [Feature Set B] [page 1281]
Custom Properties [Feature Set B] [page 14]

6.1.3.2 Manage the Account Explorer Hierarchy [Feature Set B]

Create an account structure by creating a hierarchy of directories and subaccounts using the SAP BTP cockpit. Add, move, and delete subaccounts and directories in your structure.

Context

Here, you'll learn how to:

- Add directories and subaccounts to a directory
- Move directories and subaccounts in the account structure
- Delete directories and subaccounts from the account structure

A subaccount moves with its assigned service plans and quota. The entitlements of the source and target directories are adjusted accordingly.

If any entitlement for a plan in the target directory is configured with the auto-assign option, it is then automatically applied to the moved subaccount. Automatic quota assignments are subject to available quota in the target directory.

You can create a hierarchical structure that is 7 levels deep. The highest level of a given path is always the global account and the lowest is a subaccount, which means that you can have up to 5 levels of directories.

Procedure

1. Add directories and subaccounts to a directory.
   a. When creating a new directory or subaccount, you can choose the parent directory directly in the creation dialog.
2. Move subaccounts in the account structure.
   a. Click ••• *Actions* and select *Edit*. Then select a new parent.
   b. Click ••• *Actions* and select *Move*. Then select a new parent.
   c. Click ✣ *Move* to move subaccounts by drag and dropping.
3. Delete directories and subaccounts from the account structure.
   a. Click ••• *Actions* and select *Delete*.
   b. To delete a subaccount, you can also navigate into the subaccount and click *Delete Subaccount*.

Related Information

Cloud Management Tools — Feature Set Overview
Directories [Feature Set B] [page 12]

Create a Directory [Feature Set B] [page 1277]
Create a Subaccount

6.1.3.3 Manage Users in Directories [Feature Set B]

Manage members in your directory using the SAP BTP cockpit.

Prerequisites

- You're a directory administrator.

Context

You manage directory members by assigning role collections to platform users. Use the following predefined role collections:

- Directory Administrator
- Directory Viewer

Procedure

1. In your directory, navigate to the *Users* page.
2. Click *Enable User Management* to enable the user management capabilities for this directory if the feature isn't enabled already.

   Note that only one directory in a path can manage entitlements or users.
3. Click *Edit*.
   a. To add a new user, click +, then enter the e-mail address and assigned role collection.
   b. To change a user's role collection assignment, select a new role collection in the dropdown in the *Assigned Role Collection* column.
   c. To delete a user, click 🗑
4. Click *Save*.

Related Information

Working with Users
Role Collections and Roles in Global Accounts, Directories, and Subaccounts [Feature Set B]

6.1.3.4 View Directory Usage Analytics [Feature Set B]

You can explore, compare, and analyze all your actual usage data for the services and applications that are available in your directory.

To monitor and track usage in a directory, open your directory from the *Account Explorer* in the SAP BTP cockpit, and then go to the *Usage Analytics* page in the navigation area.

> **i** Note
>
> This cockpit page is available only for directories that have user management enabled.
>
> You must also be assigned as an admin of the directory. For the more information, see Create a Directory [Feature Set B] [page 1277] and Manage Users in Directories [Feature Set B] [page 1280].

Views in the Directory 'Usage Analytics' Page

The directory *Usage Analytics* page contains two views that display the same usage information; however, your entry point is different – either from a service/subscription or a directory/subaccount:

View	Description
Service Usage	Displays high-level usage information for a selected service or business application subscriptions according to the directory from which you accessed the *Usage Analytics* page. You can also drill down and view usage data for specific subdirectories and subaccounts that are located in this directory.
Directory/Subaccount Usage	Displays high-level usage information for all services and business application subscriptions according to the directory from which you accessed the *Usage Analytics* page. You can also drill down and view usage data for specific subdirectories and subaccounts that are located in this directory.

Information is shown only for those services and subscriptions whose metered consumption in the directory is greater than zero.

Usage values are updated every 24 hours. Usage data displayed in this cockpit page refers to actual usage, not billed usage.

> → Tip
>
> - If your directory is in a global account that uses the consumption-based commercial model, you can view information about your billed usage for billable services in your global acccount's *Usage Analytics* page. The global account's *Usage Analytics* page also provides information about cloud-credit usage. For more information, see Monitoring Usage and Consumption Costs in Your Global Account.
> - You can also view usage information by subaccount directly in a subaccount's *Usage Analytics* page. For more information, see View Subaccount Usage Analytics [page 1288].

Working with the Tables and Charts

Both the *Service Usage* and *Directory/Subaccount Usage* views display the usage information in tabular and chart formats.

- The tables present accumulated usage values based on the aggregation logic of each service plan and its metric over the selected time period. The usage values are broken down by resource.
- The charts present usage values for one or more service plans that you select in the adjacent tables.

Select a table row to display the usage information in the chart. You can also compare usage between service plans in the same service by selecting multiple rows. To compare usage, the rows items must also share the same metric.

In the charts, you can view the data as a column chart or as a line chart.

To display a larger view of a chart, choose the *(Zoom)* button.

Some rounding or shortening is applied to large values. You can mouse over values in the table to view the exact values as tooltips.

Related Information

Org Administration Using the Cockpit
Cloud Management Tools — Feature Set Overview
Monitoring Usage and Consumption Costs in Your Global Account
What Is the Consumption-Based Commercial Model?
View Subaccount Usage Analytics [page 1288]
Monitoring Usage Information Using APIs of the SAP Usage Data Management Service [Feature Set B]
Manage Users in Directories [Feature Set B] [page 1280]

6.1.4 Managing Subaccounts Using the Cockpit

Learn how to structure a global account according to your organization's and project's requirements with regard to members, authorizations, and entitlements by managing subaccounts.

Context

To learn more about the concept of subaccounts, see Subaccounts.

Procedure

1. Learn how to create subaccounts.
   - Create a Subaccount
2. Learn how to manage users.
   - Managing Security Administrators in Your Subaccount [Feature Set A]
   - Add Members to Your Subaccount [Feature Set B]
3. Learn how to administer your subaccount.
   - Change Subaccount Details
   - View Subaccount Usage Analytics [page 1288]
   - Delete a Subaccount

6.1.4.1 Create a Subaccount

Create subaccounts in your global account using the SAP BTP cockpit.

Create a Subaccount [Feature Set A]

Prerequisites

You are a global account administrator.

> → Recommendation
>
> Before creating your subaccounts, we recommend you learn more about Setting Up Your Account Model.

Context

You create subaccounts in your global account. Once you create a new subaccount, you see a tile for it in the global account view, and you are automatically assigned to it as an administrator.

Procedure

1. From your global account, choose *New Subaccount*.
2. Specify a display name.
3. **Optional:** Enter a description.

4. Select the *Neo Environment* checkbox.
5. Choose the desired infrastructure provider and region for your subaccount.
6. Enter a subdomain for your subaccount. This will become part of the URL for accessing applications that you subscribe to from this subaccount.

   > **i Note**
   >
   > The subdomain can contain only letters, digits and hyphens (not allowed in the beginning or at the end), and must be unique across all subaccounts in the same region. Uppercase and lowercase letters can be used, however that alone does not qualify as a means to differentiate subdomains (e.g. `SUBDOMAIN` and `subdomain` are considered to be the same).
   >
   > The subdomain can't be changed once you have created the subaccount.

7. If your subaccount is to be used for production purposes, select the *Used for production* option.

   > **i Note**
   >
   > This does not change the configuration of your subaccount. Use this flag for your internal use to operate your production subaccounts in your global account and systems more efficiently. Your cloud operator may also use this flag to take appropriate action when handling incidents related to mission-critical accounts in production systems.
   >
   > You can change your selection at any time by editing the subaccount properties. Do not select this option if your account is used for non-production purposes, such as development, testing, and demos.

8. **Optional:** To use beta services and applications in the subaccount, select *Enable beta features*.

   > **⚠ Caution**
   >
   > You shouldn't use SAP BTP beta features in subaccounts that belong to productive enterprise accounts. For more information, see Important Disclaimers and Legal Information.

   > **i Note**
   >
   > Once you have enabled this setting in a subaccount you cannot disable it.

9. Save your changes.

Results

A new tile appears in the global account page with the subaccount details.

Next Steps

Once you've created your subaccount, navigate to it to enable the environment that you wish to use.

Create a Subaccount [Feature Set B]

Prerequisites

You are a global account administrator.

> → Recommendation
>
> Before creating your subaccounts, we recommend you learn more about Setting Up Your Account Model.

Context

You create subaccounts in your global account. Once you create a new subaccount, you see a tile for it in the global account view, and you are automatically assigned to it as an administrator.

Procedure

1. From your global account, navigate to the *Account Explorer* page.
2. Select *Create* > *Subaccount*.
3. Specify a display name.
4. **Optional:** Enter a description.
5. Enter a subdomain for your subaccount. This will become part of the URL for accessing applications that you subscribe to from this subaccount.

   > i Note
   >
   > The subdomain can contain no more than 63 characters such as letters, digits and hyphens (not allowed in the beginning or at the end), and must be unique across all subaccounts in the same region. Uppercase and lowercase letters can be used, however that alone does not qualify as a means to differentiate subdomains (e.g. `SUBDOMAIN` and `subdomain` are considered to be the same).
6. Choose the desired region for your subaccount.
7. Select a parent for your subaccount. The parent can either be the global account or a directory.
8. If your subaccount is to be used for production purposes, select the *Used for production* option.

   > i Note
   >
   > This does not change the configuration of your subaccount. Use this flag for your internal use to operate your production subaccounts in your global account and systems more efficiently. Your cloud operator may also use this flag to take appropriate action when handling incidents related to mission-critical accounts in production systems.
   >
   > You can change your selection at any time by editing the subaccount properties. Do not select this option if your account is used for non-production purposes, such as development, testing, and demos.

9. **Optional:** To use beta services and applications in the subaccount, select *Enable beta features*.

> ⚠ Caution
>
> You shouldn't use SAP BTP beta features in subaccounts that belong to productive enterprise accounts. For more information, see Important Disclaimers and Legal Information.

> ℹ Note
>
> Once you have enabled this setting in a subaccount you cannot disable it.

10. Save your changes.

Results

A new tile appears in the global account page with the subaccount details.

Next Steps

Once you've created your subaccount, navigate to it to enable the environment that you wish to use.

Related Information

Cloud Management Tools — Feature Set Overview
Global Accounts
Navigate in the Cockpit
Account Model

6.1.4.2 Change Subaccount Details

Edit subaccounts using the SAP BTP cockpit.

Context

You edit a subaccount by choosing the relevant action on its tile. It's available in the global account view, which shows all its subaccounts.

The subaccount technical name is a unique identifier of the subaccount on SAP BTP that is generated when the subaccount is created.

You can change to the following subaccount details:

Editable Subaccount Details

Field	Details
Display Name	Specify a human-readable name for your subaccount and change it later on, if necessary. This way you can distinguish between multiple subaccounts.
Description	Specify a short descriptive text about the subaccount.
Used for production	Select this option if your subaccount is being used for production purposes. This does not change the configuration of your subaccount. Use this flag for your internal use to operate your production subaccounts in your global account and systems more efficiently. Your cloud operator may also use this flag to take appropriate action when handling incidents related to mission-critical accounts in production systems. Do not select this option if your account is used for non-production purposes, such as development, testing, and demos.
Enable beta features	Enable the subaccount to use services and applications which are occasionally made available by SAP for beta usage on SAP BTP. This option is available to administrators only and is, by default, unselected. **⚠ Caution** You shouldn't use SAP BTP beta features in subaccounts that belong to productive enterprise accounts. For more information, see Important Disclaimers and Legal Information. **i Note** Once you have enabled this setting in a subaccount you cannot disable it.

Procedure

1. Choose the subaccount for which you'd like to make changes and choose  on its tile.

   You can view more details about the subaccount such as its description and additional attributes by choosing *Show More*.
2. Make your changes and save them.

Related Information

Using Beta Features with Subaccounts [page 11]

6.1.4.3 View Subaccount Usage Analytics

You can explore, compare, and analyze all your actual usage data for the services and applications that are available in your subaccount.

To monitor and track usage in a subaccount, open the subaccount in the SAP BTP cockpit and choose *Usage Analytics* in the navigation area.

Views in the Subaccount 'Usage Analytics' Page

The subaccount *Usage Analytics* page contains views that display usage at different levels of detail:

View	Description
Subaccount	Displays high-level usage information for your subaccount relating to services and business application subscriptions. Some information in this view is displayed only for global account admins.
Services	Displays usage per service plan for the region of the subaccount, and the selected metric and period. Information is shown for all services whose metered consumption in the subaccount is greater than zero.

> ### i Note
>
> The information displayed in this page depends on the environments that are enabled for your subaccount. For example, information about spaces is displayed only for subaccounts in the Cloud Foundry environment.

Usage values are updated every 24 hours.

Usage data displayed in this cockpit page refers to actual usage, not billed usage.

> ### → Tip
>
> - [Feature Set A] If your subaccount is in a global account that uses the consumption-based commercial model, you can view information about your billed usage for billable services in your global acccount's *Overview* and subaccount's *Overview* pages. The global account's *Overview* page also provides information about cloud-credit usage.
> - [Feature Set B] If your subaccount is in a global account that uses the consumption-based commercial model, you can view information about your billed usage for billable services in your global acccount's *Usage Analytics* page. The global account's *Usage Analytics* page also provides information about

cloud-credit usage. A directory's *Usage Analytics* page also allows you to view usage information for each subaccount that is located in the directory.

- [Feature Set B] If you use directories to group your subaccounts, you can view usage information by directory in your global account's *Usage Analytics* page or directly in the directory's *Usage Analytics* page.

For more information, see Monitoring Usage and Consumption Costs in Your Global Account and View Directory Usage Analytics [Feature Set B] [page 1281].

Working with the Tables and Charts

In the *Services* view, the usage information is displayed in both tabular and chart formats.

- The tables present accumulated usage values based on the aggregation logic of each service plan and its metric over the selected time period. The usage values are broken down by resource.
- The charts present usage values for one or more service plans or spaces that you select in the adjacent tables. The resolution of the charts is automatically set to days, weeks, or months, depending on the range of the selected time period.

You can perform various actions within the tables and charts:

- In the *Services* view, select a table row to display the usage information in the chart. You can also select multiple rows to compare usage between service plans in the same service. Multi-row selection in the table is possible only when you have selected a single service in the *Service* filter and the row items share the same metric.
- In the charts, you can view the data as a column chart or as a line chart.
- To display a larger view of a chart, choose the *(Zoom)* button.
- Some rounding or shortening is applied to large values. You can mouse over values in the table to view the exact values as tooltips.

Using the Filters

Use the filters in the *Services* view to choose which usage information to display.

You can apply the *Metric* filter only when you have selected a service with more than one metric.

Click the *(Reset)* icon to reset the filters to their default settings.

Related Information

Org Administration Using the Cockpit
Cloud Management Tools — Feature Set Overview
Monitoring Usage and Consumption Costs in Your Global Account
What Is the Consumption-Based Commercial Model?

View Directory Usage Analytics [Feature Set B] [page 1281]
Monitoring Usage Information Using APIs of the SAP Usage Data Management Service [Feature Set B]

6.1.4.4 Delete a Subaccount

Delete subaccounts using the SAP BTP cockpit.

Prerequisites

- You're a global account administrator.
- You're a subaccount security administrator.
- The subaccount doesn't contain any active subscriptions, service instances, brokers, or platforms.

Context

To prevent accidental deletion of subaccounts and creation of orphaned data, any active subscriptions, service instances, brokers, or platforms must be removed from the subaccount before it can be deleted. Only subaccount administrators can remove such content from a subaccount.

Procedure

1. Navigate into the subaccount you want to delete.
2. Choose the subaccount that you want to delete.
3. Choose *Delete Subaccount* and confirm the operation.

Related Information

Relationship Between Global Accounts and Subaccounts [page 13]

6.1.5 Managing Entitlements and Quotas Using the Cockpit

When you purchase an enterprise account, you are entitled to use a specific set of resources, such as the amount of memory that can be allocated to your applications.

An entitlement equals your right to provision and consume a resource. A quota represents the numeric quantity that defines the maximum allowed consumption of that resource.

Entitlements and quotas are managed at the global account level, distributed to subaccounts, and consumed by the subaccounts. When quota is freed at the subaccount level, it becomes available again at the global account level.

> **i Note**
>
> Only global account administrators can configure entitlements and quotas for subaccounts.

There are two places in the SAP BTP cockpit where you can view and configure entitlements and quotas for subaccounts - at global account level and at subaccount level. Depending on your permissions, you may only have access to one of these pages. You can find more details below:

Entitlements Pages [Feature Set A]

Page in cockpit	Navigation Level	Visible to	Permissions
Entitlements > *Subaccount Assignments*	Global account level	Global account administrators only	• View & Edit - Global account administrators
Entitlements > *Service Assignments*	Global account level	Global account administrators only	• View - Global account administrators (you cannot make changes to entitlements or quota assignments from this page)
Entitlements	Subaccount level	Subaccount members (regardless of whether they are also global account administrators or not)	• View - Subaccount members who are not global account administrators • View & Edit - Global account administrators who are also subaccount members

You can also assign entitlements to directories, see Configure Entitlements and Quotas for Directories [Feature Set B].

Related Information

Configure Entitlements and Quotas for Subaccounts [page 1292]
Add Quotas to Subaccounts Using the Console Client [page 1347]
Troubleshooting: Entitlement and Quota Problems

6.1.5.1 Configure Entitlements and Quotas for Subaccounts

Assign entitlements to subaccounts by adding service plans and distribute the quotas available in your global account to your subaccounts using the SAP BTP cockpit.

Prerequisites

You must be a global account administrator to configure subaccount entitlements.

Context

You can distribute entitlements and quotas across subaccounts within a global account from two places in the cockpit:

- The *Entitlements* > *Subaccount Assignments* page at global account level (only visible to global account administrators)
- The *Entitlements* page at subaccount level (visible to all subaccount members, but only editable by global account administrators)

To get an overview of all the services and plans available in the global account, you can navigate to *Entitlements* > *Service Assignments* using the left hand-side navigation. There you can see the global usage of each service plan, as well as the detailed assignments of each service across subaccounts, but you aren't able to make changes on this page.

For more information, see Managing Entitlements and Quotas Using the Cockpit [page 1291].

Configure Entitlements and Quotas from Your Global Account

Procedure

1. Navigate to your global account.
2. Choose *Entitlements* > *Subaccount Assignments* from the left hand-side navigation.
3. At the top of the page, select all the subaccounts for which you would like to configure or display entitlements.

   > → Tip
   >
   > If your global account contains more than 20 subaccounts, choose to open up the value help dialog. There you can filter subaccounts by role, environment, and region to make your selection easier and faster. You can only select a maximum of 50 subaccounts at once.

4. Choose *Go* to apply the filter.

   You get a table for each of the subaccounts you selected, displaying the current entitlement and quota assignments.

5. Choose *Configure Entitlements* to start editing entitlements for a particular subaccount.

   > **i Note**
   >
   > You can only edit entitlements for one subaccount at a time.

6. You can now edit the entitlements table:

Action	Steps
Add new service plans to the subaccount	Choose *Add Service Plans* and from the dialog select the services and the plans from each service that you would like to add to the subaccount.
Edit the quota for one or more service plans	Use + and − to increase or decrease the quota for each service plan.
Assign quota to unlimited service plans in global accounts using the consumption-based commercial model.	Use the checkbox in the *Assign Quota* column to enable or disable the *Subaccount Assignment* column for editing. Now, you can increase or decrease the quota for this service plan by using + and −.
Delete a service plan and its quota from the subaccount	Choose 🗑 from the *Actions* column.

7. Once you're done, choose *Save* to save the changes and exit edit mode for that subaccount.
8. **Optional:** Repeat steps 5 to 7 to configure entitlements for the other subaccounts selected.

Configure Entitlements and Quotas from Your Subaccount

If you're working in a subaccount and realize you're missing entitlements or quota, you can edit the entitlements directly from that subaccount.

Prerequisites

In addition to being a global account administrator, you must also be a member of the subaccount to be able to access that subaccount.

Procedure

1. Navigate into the subaccount where you would like to configure the entitlements.
2. Choose *Entitlements* from the left hand-side navigation to see the entitlements for your subaccount.

3. Choose *Configure Entitlements*.
4. You can now edit the entitlements table:

Action	Steps
Add new service plans to the subaccount	Choose *Add Service Plans* and from the dialog select the services and the plans from each service that you would like to add to the subaccount.
Edit the quota for one or more service plans	Use + and − to increase or decrease the quota for each service plan.
Delete a service plan and its quota from the subaccount	Choose 🗑 from the *Actions* column.

5. Once you're happy with the changes, choose *Save* to save and exit edit mode.

6.1.6 Managing Members Using the Cockpit

You can add members to your global accounts and subaccounts and assign different roles to them:

- Members and Roles in the Neo Environment [page 1294]
- Add Members to Your Global Account [page 1297]
- Add Members to Your Neo Subaccount [page 1299]
- Managing Member Authorizations in the Neo Environment [page 1301]

6.1.6.1 Members and Roles in the Neo Environment

A member is a user who is assigned to an SAP BTP global account or subaccount. A member automatically has the permissions required to use the SAP BTP functionality within the scope of the respective accounts and as permitted by their account member roles.

Roles

Roles determine which functions in the cockpit users can view and access, and which actions they can initiate.

Roles support typical tasks performed by users when interacting with the cloud platform, for example, adding and removing users. A user can be assigned one or more roles, where each role comes with a set of permissions. The set of assigned roles defines what functionality is available to the user and what activities he can perform.

The Administrator role in a global account is automatically assigned to the user who has purchased resources for an enterprise account. A global account administrator has permissions to manage all roles in the relevant global account and can assign the Administrator role to other users in the same global account. He can add members to the global account who are then automatically assigned the Administrator role.

On the *Members* page at the global account level in the cockpit, all global account members can view the global account administrators.

Note the following:

- Users can be assigned to one or more subaccounts and to one or more roles in the relevant subaccount.
- If the user is assigned to more than one subaccount, an administrator must assign the roles to the user for each subaccount.
- Roles apply to all operations associated with the subaccount, irrespective of the tool used (Eclipse-based tools, cockpit, and console client).
- As an administrator in the Neo environment, you cannot remove your own administrator role. You can remove any member except yourself.

6.1.6.1.1 SAP ID Service

The default platform identity provider and application identity provider of SAP BTP is SAP ID service.

Trust to SAP ID service in your subaccount is pre-configured in SAP BTP by default, so you can start using it without further configuration. Optionally, you can add additional trust settings or set the default trust to inactive, for example if you prefer to use another SAML 2.0 identity provider. Using the SAP BTP cockpit you can make changes by navigating to your respective subaccount and choosing *Security* > *Authorization*.

If you want to add new users to a subscribed app, or if you want to add users to a service, such as Web IDE, you can add those users to SAP ID service in your subaccount. See Add Users to SAP ID Service in the Neo Environment [page 1295].

> **i Note**
>
> If you want to use a custom IdP, you must establish trust to your custom SAML 2.0 identity provider. We describe a custom trust configuration using the example of SAP Cloud Identity Services - Identity Authentication.
>
> For more information, see Trust and Federation with Identity Providers.

6.1.6.1.1.1 Add Users to SAP ID Service in the Neo Environment

Before you can assign roles or role collections to a user in SAP ID service, you have to ensure that this user is assigned to SAP ID service.

Prerequisites

The user you want to add to SAP ID service must have an SAP user account (for example, an S-user or P-user). For more information, see Create SAP User Accounts .

Context

Procedure

1. Go to your subaccount and choose *Security* > *Authorizations*.
2. Choose the trust configuration for SAP ID service.
3. Enter the business user's name in the *User* field, for example `john.doe@example.com`.

> → Remember
>
> If the user identifier you entered does not have an SAP user account or has never logged on to an application in this subaccount, SAP BTP cannot automatically verify the user name. To avoid mistakes, you must ensure that the user name is correct and that it exists in SAP ID service.

Related Information

Platform Identity Provider [page 1746]

6.1.6.1.1.2 Create SAP User Accounts

If you want to grant authorizations to users from SAP ID service in your subaccount, you must ensure that they have a user account in SAP ID service.

Context

Procedure

Choose one of the following ways to create SAP user accounts:

- Use the user management of SAP One Support Launchpad to create S-users.
- In a trial account, use self-registration.
  - Register directly at SAP ID Service.
  - Register at SAP Developer Center.

6.1.6.2 Add Members to Your Global Account

Add users as global account members using the SAP BTP cockpit.

Prerequisites

You have the Administrator role in the global account.

Context

Procedure

1. Find out which cloud management tools feature set your global account uses. For more information, see Cloud Management Tools — Feature Set Overview.
2. Learn how to add members to your global account.
   - Add Members to Your Global Account [Feature Set A] [page 1297]
   - Add Members to Your Global Account [Feature Set B] [page 1298]

Related Information

Cloud Management Tools — Feature Set Overview
User and Member Management

Add Members to Your Global Account [Feature Set A]

Add users as global account members using the SAP BTP cockpit.

Context

The users you add as members at global account level are automatically assigned the Administrator role.

All global account administrators can do the following:

- View all the subaccounts in the global account, meaning all the subaccount tiles in the global account's *Subaccounts* page.
- Edit general properties of the subaccounts in the global account from the *Edit* icon in the subaccount tile.
- Create a new subaccount in the global account.
- View, add, and remove global account members.
- Manage entitlements for the global account.

By default, the cockpit and console client are configured to use SAP ID Service, which uses the SAP user base, as an identity provider for user authentication. If you want to use a custom user base and custom Identity Authentication tenant settings, see Platform Identity Provider [page 1746].

On the *Members* page at the global account level in the cockpit, all global account members can view the global account administrators. If you are a subaccount member, you are also a member of the global account.

Procedure

1. Choose the global account to which you'd like to add members.
2. In the navigation area, choose *Members*.
3. Choose *Add Members*.

   If you want to use a custom user base, choose *Other* for *User Base* and then enter the corresponding Identity Authentication tenant name. For more information, see Platform Identity Provider [page 1746].
4. Enter up to 100 user IDs. To separate them, you can use commas, spaces, semicolons, or line breaks.
5. Choose *Add Members*.

Next Steps

To delete members at global account level, choose 🗑 *(Delete)* next to the user's ID.

Add Members to Your Global Account [Feature Set B]

Add members to your global account by assigning them a predefined role collection.

Prerequisites

- You must be a global account administrator in order to add other global account members.
- You have defined role collections. For more information, see Define a Role Collection.

Context

There are 2 predefined role collections that you can use when adding global account members:

- Global Account Administrator
- Global Account Viewer

For more information on the roles included in each of these role collections, see Role Collections and Roles in Global Accounts, Directories, and Subaccounts [Feature Set B].

Procedure

1. Familiarize yourself with the concepts of Working with Role Collections.
2. Navigate to your global account.
3. Assign a role collection to users. For more information, see Assign Users to Role Collections.

Results

The new role collection assigned to the user is displayed in the table. The user is now a global account member.

6.1.6.3 Add Members to Your Neo Subaccount

Add users as members to a subaccount in the Neo environment and assign roles to them using the SAP BTP cockpit.

Prerequisites

- You have an enterprise account.
- You have the Administrator role for the subaccount.
- You have the user IDs of the members that you want to add.

> → Tip
>
> *User Management* application: 1271482.

Context

In the Neo environment, you can assign predefined roles to subaccount members, but you can also create custom platform roles. For more information, see Manage Custom Platform Roles [page 1306]. Administrators

can request S-user IDs on the SAP ONE Support Launchpad Managing Member Authorizations in the Neo Environment [page 1301].

Procedure

1. Choose the subaccount to which you'd like to add members.
2. In the navigation area, choose *Members*.
   Administrators can request S-user IDs on the SAP ONE Support LaunchpadAll members currently assigned to the subaccount are shown in a list.

   > **i Note**
   >
   > The name of a member is shown only after he or she visits the subaccount for the first time.

3. Choose *Add Members*.
4. Enter one or more user IDs, separated by commas, spaces, semicolons, or line breaks.
   There is currently no user validation.
5. Select the roles for the members and save your changes.

Next Steps

- To select or unselect roles for a member, choose  (Edit). The changes you make to the member's roles take effect immediately.
- You can enter a comment when editing user roles. This lets you track the reasons for subaccount membership and other important data. The comments are visible to all members.
- You can send an e-mail to a member. This option appears only after the recipient visits the subaccount for the first time.
- To remove all the roles of a member, choose *Delete* (trashcan). This also removes the member from the subaccount.
- Choose the *History* button to view a list of changes to members (for example, added or removed members, or changed role assignments).
- Use the filter to show only the members with the role you've selected.

Related Information

Managing Member Authorizations in the Neo Environment [page 1301]

6.1.6.4 Managing Member Authorizations in the Neo Environment

SAP BTP includes predefined platform roles that support the typical tasks performed by users when interacting with the platform. In addition, subaccount administrators can combine various scopes into a custom platform role that addresses their individual requirements.

A platform role is a set of permissions, or scopes, managed by the platform. Scopes are the building blocks for platform roles. They represent a set of permissions that define what members can do and what platform resources they can access (for example, configuration settings such as destinations or quotas). Most scopes follow a "Manage" and "Read" pattern. For example, `manageXYZ` comprises the actions create, update, and delete on platform resource XYZ. However, some areas use a different pattern, for example, Application Lifecycle Management.

Predefined Platform Roles

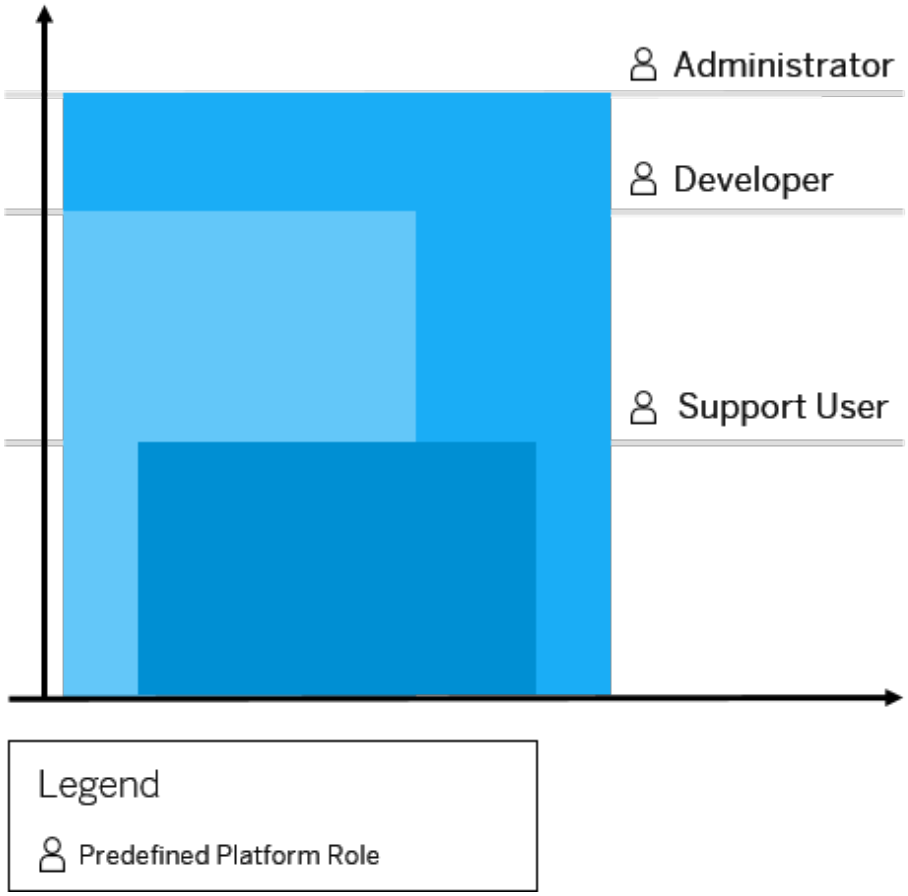
Predefined platform roles cannot be changed. However, global account administrators can copy from predefined roles, and then modify the copies.

SAP BTP includes the following predefined platform roles:

Role	Description
Administrator	Manage subaccount members You can also manage subscriptions, trust, authorizations, and OAuth settings, and restart SAP HANA services on HANA databases. Furthermore, you can view heap dumps and download a heap dump file. In addition, you have all permissions granted by the developer role, except the debug permission. **i Note** This role also grants permissions to view the *Connectivity* tab in the SAP BTP cockpit.
Cloud Connector Admin	Open secure tunnels via Cloud Connector from on-premise networks to your subaccounts. **i Note** This role also grants permissions to view the *Connectivity* tab in the SAP BTP cockpit.

Role	Description
Developer	Supports typical development tasks, such as deploying, starting, stopping, and debugging applications. You can also change loggers and perform monitoring tasks, such as creating availability checks for your applications and executing MBean operations. **i Note** By default, this role is assigned to a newly created user.
Support User	Designed for technical support engineers, this role enables you to read almost all data related to a subaccount, including its metadata, configuration settings, and log files. For you to read database content, a database administrator must assign the appropriate database permissions to you.
Application User Admin	Assigned by the subaccount administrator to a subaccount member. Manage user permissions on application level to access Java, HTML5 applications, and subscriptions. You can control permissions directly by assigning users to specific application roles or indirectly by assigning users to groups, which you then assign to application roles. You can also unassign users from the roles or groups. **i Note** This role does not let you manage subaccount roles and perform actions at the subaccount level (for example, stopping or deleting applications).

The following graphic illustrates the predefined Administrator, Developer, and Support User roles and their amount of scopes:



Amount of Platform Scopes

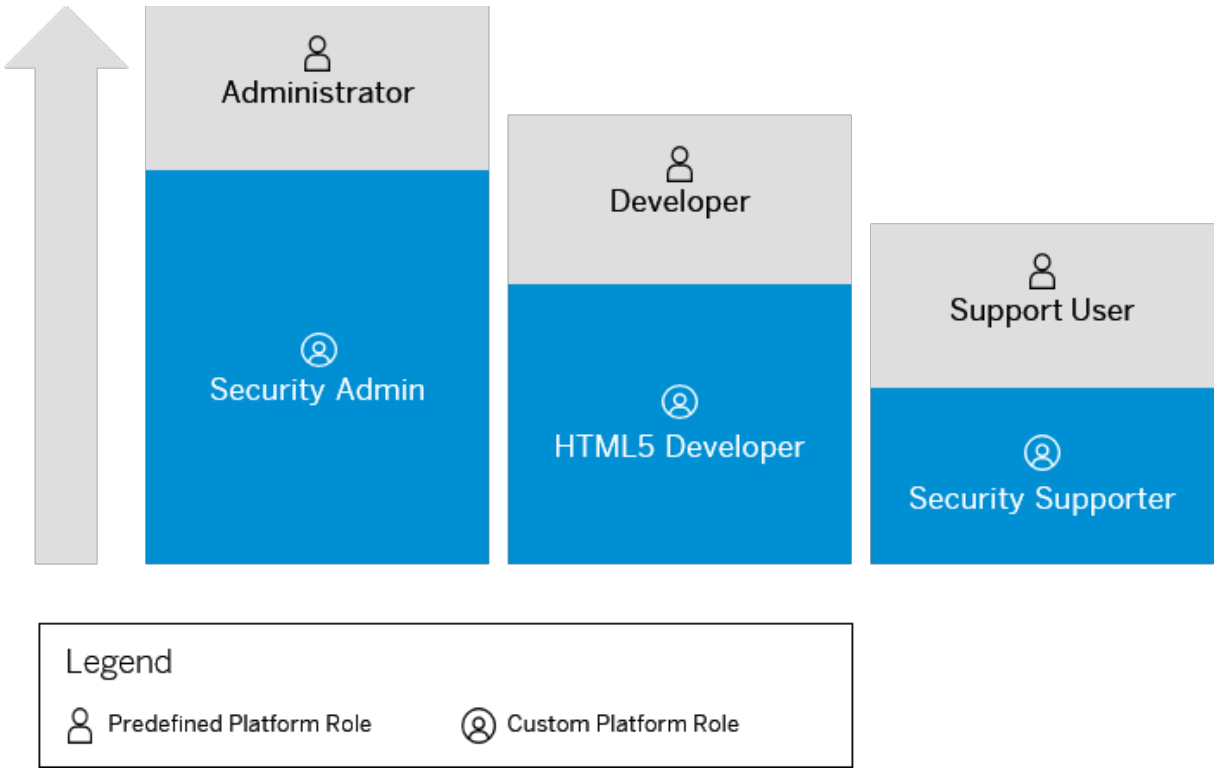
The Admin role includes all platform scopes available on SAP BTP. The Developer and Support User are subsets of the Admin role.

Custom Platform Roles

Administrators of a subaccount can define custom platform roles based on their needs by assembling the different scopes they want their custom platform role to include. Custom platform roles are managed at subaccount level and can be changed at any time.

Subaccount administrators can combine various scopes into a custom platform role that addresses their individual requirements. Scopes are the building blocks for platform roles. They represent a set of permissions that define what members can do and what platform resources they can access (for example, configuration settings such as destinations or quotas).

The following example illustrates how custom platform roles in SAP BTP typically look like regarding their amount of scopes:



Related Information

Platform Scopes [page 1307]
Manage Custom Platform Roles [page 1306]

6.1.6.4.1 Enable Application Providers to Access Your Subaccount

If your scenario requires it, you can add application providers as members to your SAP BTP subaccount in the Neo environment and assign them the administrator role so that they can deploy and administer the applications you have purchased.

Prerequisites

- You have an enterprise account.
- You have the Administrator role for the subaccount.
- You have the user ID of the user responsible for deploying and managing the applications you have purchased. You can get this information from your application provider.

> → Tip
>
> You can request user IDs at the SAP Service Marketplace: http://service.sap.com/request-user
>
> SAP Service Marketplace users are automatically registered with the SAP ID service, which controls user access to SAP BTP.

Context

As an administrator of a subaccount, you can add members to it and make them administrators of the subaccount using the SAP BTP cockpit. For example, if you have purchased an application from an SAP implementation partner, you may need to enable the partner to deploy and administer the application.

Procedure

1. Choose the subaccount to which you'd like to add members.
2. In the navigation area, choose *Members*.
   All members currently assigned to the subaccount are shown in a list.
3. Choose *Add Members*.
4. Enter the user IDs you have received from your application provider, separated by commas, spaces, semicolons, or line breaks.

   User IDs are case-insensitive and can contain alphanumeric characters only. Currently, there is no user validation.
5. Select the *Administrator* checkbox.

   > ℹ Note
   >
   > By default, the *Developer* option is selected. Do not unselect this checkbox.
6. Choose *Add Members*.
   Any changes you make take effect immediately. The users are added to the list of team members and are assigned the developer and the administrator role. They can now deploy and administer applications in your subaccount.

   > ℹ Note
   >
   > You cannot remove your own administrator role.
7. Notify your application provider that they now have the necessary permissions to access the subaccount.

6.1.6.4.2 Manage Custom Platform Roles

Subaccount administrators can define custom platform roles and assign them to the members of its subaccounts.

Prerequisites

You have the Administrator role for the subaccount.

Context

Procedure

1. Choose *Platform Roles* in the navigation area for the subaccount for which you'd like to manage custom platform roles.
   All custom and predefined platform roles available for the subaccount are shown in a list.
2. You have the following options:
   - Create, edit, and delete custom platform roles.
   - Create a copy of an existing custom platform role and make changes to your copy.
   - View custom and predefined platform role details, including the scopes assigned to it, in read-only mode.

     > **i Note**
     >
     > You cannot change or delete a predefined platform role, but you can copy from it and make changes to the copy.

   - Use the search function:
     - You can search for all the elements in the list of platform roles and restrict it to the entries you are interested in.
     - In the Create, Copy, and the Edit dialogs, find the scopes that you want to combine into the custom platform role quickly.
3. To create a custom platform role, choose *New Platform Role*.
   a. Enter an ID (mandatory), display name, and a description.
   b. Select the platform scopes and save your changes.
4. To create a custom or a predefined platform role by copying from an existing one, choose *(Copy)* and proceed as in the previous step. The scopes from the original role are selected by default.
5. To edit a custom platform role, choose *(Edit)*.
6. To view platform role details, including the scopes assigned to it, in read-only mode, choose *(Display)*.

7. To delete a custom platform role, choose  *(Delete)*.
8. To assign a custom platform role to a member, perform the steps described in Add Members to Your Neo Subaccount [page 1299].

Related Information

Platform Scopes [page 1307]
Managing Member Authorizations in the Neo Environment [page 1301]

6.1.6.4.3 Platform Scopes

Category	ID	Scope Name	Description
Account Management	readAccount	View Accounts	Enables you to view a list of all subaccounts available to you and access them.
	manageAccount	Manage Accounts	Enables you to edit and delete subaccounts.
	readQuota (obsolete)	View Quota	Enables you to view the purchased quotas in use and how they are distributed between individual subaccounts.
	manageQuota (obsolete) Only global account administrators can manage quota.	Manage Quota	Enables you to distribute free quota and reassign quota between individual subaccounts.
	readCustomPlatformRoles	View Custom Platform Roles	Enables you to list self-defined platform roles.
	manageCustomPlatformRoles	Manage Custom Platform Roles	Enables you to define your own platform roles.
	manageAccountNotifications	Manage Account Notifications	Enables you to further process a notification, for example, you can assign it to yourself or set the status to complete.

Category	ID	Scope Name	Description
	readAccountNotifications	Read Account Notifications	Enables you to read notifications for a subaccount.
Agent Activation for Dynatrace Service	readDynatraceIntegration	Manage Dynatrace Integration	Enables you to view the configuration settings of the Agent Activation for Dynatrace service in your subaccount.
	manageDynatraceIntegration	Read Dynatrace Integration	Enables you to update and delete the configuration settings of the Agent Activation for Dynatrace service in your subaccount.
Authorization Management	readApplicationRoles	View Application Roles	Enables you to list all user roles available for a Java application.
	manageApplicationRoles	Manage Application Roles	Enables you to assign user roles for a Java application and create new roles.
	readAuthorizationSettings	View Authorization Settings	Enables you to view all kinds of role, group, and user mappings on account and subscription level. **i Note** Assigning this scope to a role requires to assign the readAccount and readSubscriptions scopes as well.
	manageAuthorizationSettings	Manage Authorization Settings	Enables you to manage all kinds of role, group, and user mappings on account and subscription level. **i Note** Assigning this scope to a role requires to assign the readAccount, readSubscriptions and readAuthorizationSettings scopes as well.
Connectivity Service	readDestinations	View Destinations	Enables you to view destinations required for communication outside SAP BTP.

Category	ID	Scope Name	Description
	manageDestinations	Manage Destinations	Enables you to create, update, and delete destinations and their properties required for communication outside SAP BTP.
	readSCCTunnels	View SCC Tunnels	Enables you to view the data transmission tunnels used by the Cloud connector to communicate with backend systems.
	manageSCCTunnels	Manage SCC Tunnels	Enables you to operate the data transmission tunnels used by the Cloud connector.
Document Service	listECMRepositories	List Document Service Repositories	Enables you to list the document service repositories.
	manageECM	Manage Document Service Repositories	Enables you to create and delete document service repositories, including the management of repository keys.
Enterprise Messaging	readMessagingService	Read Messaging Service	Enables you to view details of messaging hosts, queues, and applications bindings to messaging hosts.
	manageMessagingQueuesAndBindings	Manage Queues and Application Bindings	Enables you to: • Create, edit, and delete queues. • Clear all messages from a queue. • Create and delete application bindings.
	manageMessagingHosts	Manage Messaging Hosts	Enables you to create, edit, and delete messaging hosts.
Extension Integration Management	readExtensionIntegration	Read Extension Integration	Enables you to read integration tokens.
	manageExtensionIntegration	Manage Extension Integration	Enables you to create and delete integration tokens.
	manageApplicationRoleProvider	Manage Application Role Provider	Enables you to manage Java applications' role provider using the SAP BTP cockpit.
	readApplicationRoleProvider	Read Application Role Provider	Enables you to read the Java applications' role provider configuration using the SAP BTP cockpit.

Category	ID	Scope Name	Description
	readExtensionConfiguration	Read Extension Configuration	Enables you to read the extension configuration.
	manageExtensionConfiguration	Manage Extension Configuration	Enables you to manage the extension configuration.
Git Service	accessGit	Access Git Repositories	Enables you to create repositories, push commit, push tags, create new remote branches, and push commits authored by other users (forge author identity). **i Note** Assigning this scope to a role requires to assign the readHTML5Applications scope as well.
	manageGit	Manage Git Repositories	Enables you to delete repositories, run garbage collection on repositories, lock and unlock repositories, delete remote branches, delete tags, push commits made by other users (forge committer identity), forcefully push commits, for example to rewrite the history of a Git repository, and forcefully push tags, for example to move the version of an HTML5 application to a different commit. **i Note** Assigning this scope to a role requires to assign the readHTML5Applications scope as well.
	readGit	Browse Git Repositories	Enables you to clone a repository as well as to fetch commits and tags. **i Note** Assigning this scope to a role requires to assign the readHTML5Applications scope as well.

Category	ID	Scope Name	Description
HTML5 Application Management	readHTML5Applications	List HTML5 Applications	Enables you to list HTML applications and review their status. **i Note** Assigning this scope to a role requires to assign the readAccount scope as well.
	manageHTML5Applications	Manage HTML5 Applications	Enables you to create, start, update, and delete HTML5 applications. **i Note** Assigning this scope to a role requires to assign the readAccount scope as well.
Java Application Lifecycle Management	readJavaApplications	List Java Applications	Enables you to list Java applications, get their status, and list Java application processes. **i Note** Assigning this scope to a role requires to assign the readMonitoringData scope as well.
	manageJavaApplications	Manage Java Applications	Enables you to create, deploy, start, stop, update, and delete a Java application. **i Note** Assigning this scope to a role requires to assign the readMonitoringData scope as well.
	manageJavaProcesses	Manage Java Processes	Enables you to start or stop Java application processes. **i Note** Assigning this scope to a role requires to assign the readMonitoringData scope as well.

Category	ID	Scope Name	Description
	profileApplication	Profile Java Applications	Enables you to profile Java applications.
	debugApplication	Debug Java Applications	Enables you to debug Java applications.
Keystore Service	manageKeystores	Manage Keystores	Enables you to manage (create, delete, list) key stores (using the console client).
	readKeystores	View Keystores	Enables you to lists all the services grouped by service category in the cockpit and view their enablement status.
Logging Service	readLogs	View Application Logs	Enables you to view all logs available for a Java application.
	manageLogs	Manage Application Logs	Enables you to change the log level for Java application logs.
	readHeapDumps	View Heap Dumps	Enables you to view heap dumps as well as to download heap dump files for a Java application. **i Note** Assigning this scope to a role requires to assign the readLogs scope as well.
Member Management	readAccountMembers	View Account Members	Enables you to view a list of members for an individual subaccount. **i Note** Assigning this scope to a role requires to assign the readCustomPlatformRoles scope as well.
	manageAccountMembers	Manage Account Members	Enables you to add and remove members for an individual subaccount and to assign user roles to them.
Metering Service	readMeteringData	Read Metering Data	Enables you to access data related to your application's resource consumption, e.g. network data volume or database size.

Category	ID	Scope Name	Description
Monitoring Service	readMonitoringConfiguration	Read Monitoring Configuration	Enables you to list JMX checks, availability checks, and alert recipients.
	readMonitoringData	Read Monitoring Data	Enables you to view history of metrics, current metrics, and JMX MBeans.
	manageMonitoringConfiguration	Manage Monitoring Configuration	Enables you to set and update JMX checks, availability checks, and alert recipients. It also allows you to manage custom HTTP checks.
	executeJMXOperation	Execute JMX Operation	Enables you to run JMX operations through the JMX console.
Multi-Target Application Management	readMultiTargetApplication	Browse Solutions Inventory	Enables you to list solutions, get their status, and list solution operations.
	manageMultiTargetApplication	Manage Solutions	Enable you to deploy, delete, and subscribe to solutions using Multi-Target Applications (MTA).
OAuth Client Management	readOAuthSettings	View OAuth Settings	Enables you to view OAuth Application Client settings. **i Note** This scope is used for viewing the OAuth clients for **OAuth-protected applications**, not for the platform APIs. Each platform API requires its own scopes, described in its documentation. **i Note** Assigning this scope to a role requires to assign the readAccount and readSubscriptions scopes as well.

Category	ID	Scope Name	Description
	manageOAuthSettings	Manage OAuth Settings	Enables you to manage OAuth Application Client settings. **i Note** This scope is used for managing the OAuth clients for **OAuth-protected applications**, not for the platform APIs. Each platform API requires its own scopes, described in its documentation. **i Note** Assigning this scope to a role requires to assign the readAccount, readSubscriptions and readOAuthSettings scopes as well.
Password Storage	managePasswords	Manage Passwords	Enables you to set and delete passwords for a given application in the password storage (using the console client).
	readPasswords	Read Passwords	Enables you to retrieve the passwords put into password storage for a given application (using the console client).
SAP HANA / SAP ASE Service	readDatabaseInformation	View Database Information	Enables you to view lists of SAP HANA and SAP ASE database systems, databases, and database-related service requests. You can also view information such as the assigned database type, the database version, and data source bindings.

Category	ID	Scope Name	Description
	administrateDatabases	Administrate Databases	Enables you to manage the lifecycle of SAP HANA and SAP ASE database systems (upgrade, restart, etc.). It also allows you how to create SAP HANA database users and assign the administrator role to them (not applicable to SAP HANA MDC tenant databases). In addition, it enables you to allow members of other subaccounts to open tunnels to SAP HANA and SAP ASE databases in your subaccount.
	manageDatabases	Manage Databases	Enables you to manage SAP HANA and SAP ASE databases, database schemas for shared SAP HANA databases, and data source bindings. It also allows you to create database users and assign the developer role to them (only applicable to SAP ASE databases and shared SAP HANA databases).
	openDatabaseTunnels	Manage Database Tunnels	Enables you to open tunnels to SAP HANA and SAP ASE databases.
Service Management	listServices	List Services	Enables you to browse through the list of services and review their status in your subaccount.
	manageServices	Manage Services	Allows you to list and enable services in the cockpit. **i Note** For applying any service-specific configuration, additional scopes (e.g. manageDestinations) may be required.

Category	ID	Scope Name	Description
Subscription Management	manageSubscriptions	Manage Subscriptions	Enables you to manage subscribed Java and HTML5 applications. **i Note** Assigning this scope to a role requires to assign the manageHTML5Applications scope as well. It is planned to remove this requirement in a future release.
	readSubscriptions	View Subscriptions	Enables you to list your subscribed Java and HTML5 applications. **i Note** Assigning this scope to a role requires to assign the readHTML5Applications scope as well. It is planned to remove this requirement in a future release.
Trust Management	readTrustSettings	View Trust Settings	Enables you to read trust configurations. **i Note** Assigning this scope to a role requires to assign the readAccount, readAuthorizationSettings and readSubscriptions scopes as well.
	manageTrustSettings	Manage Trust Settings	Enables you to manage trust configurations (change local provider, add trusted application/platform IdPs). **i Note** Assigning this scope to a role requires to assign the readTrustSettings, readAccount, readAuthorizationSettings and readSubscriptions scopes as well.

Category	ID	Scope Name	Description
Virtual Machines	readVirtualMachines	List Virtual Machines	Enables you to list virtual machines, volumes, volume snapshots, security group rules, to get their status, and list virtual machine processes.
	manageVirtualMachines	Manage Virtual Machines	Enables you to create and delete virtual machines, volumes, volume snapshots, security group rules, and access points.

6.2 Account Administration Using the SAP BTP Command Line Interface (btp CLI) [Feature Set B]

Use the SAP BTP command line interface (btp CLI) for account management tasks, such as creating or updating subaccounts and directories, and managing entitlements. It is a an alternative to the SAP BTP cockpit for all users who like to work in a terminal or want to automate operations using scripts.

The btp CLI offers convenient features for interactive use as well as for scripting: For the interactice user, there's login via SSO, command autocompletion, as well as several interactive commands with prompts. But all commands can also be run with parameters, so the btp CLI can just as well be used for scripting. In addition, there is the `--format JSON` option to receive the results as JSON instead of text, and it exposes exit codes for the result status. You can then use the capabilities of the shell, e.g. jq, to work with the results.

For a basic tutorial about automation with btp CLI, see Automate Account Operations with the Command Line Interface.

> **Note**
>
> With the release of version 2.0.0 on March 25, 2021, the executable file of the CLI client was renamed from **sapcp** to **btp**. All commands remain compatible, and we will support the sapcp CLI until September 2021. However, this documentation and all examples inside refer to the **btp CLI**, and we recommend to download the latest btp CLI client from the SAP Development Tools page. See Migrating from sapcp to btp [page 1322].

> **Note**
>
> The content in this section is only relevant for cloud management tools feature set B. For more information, see Working with Cloud Management Tools Feature Set B in the Neo Environment [page 819] and Cloud Management Tools - Feature Set Overview.

Learn How to Work With the btp CLI

Learn how to use the client:

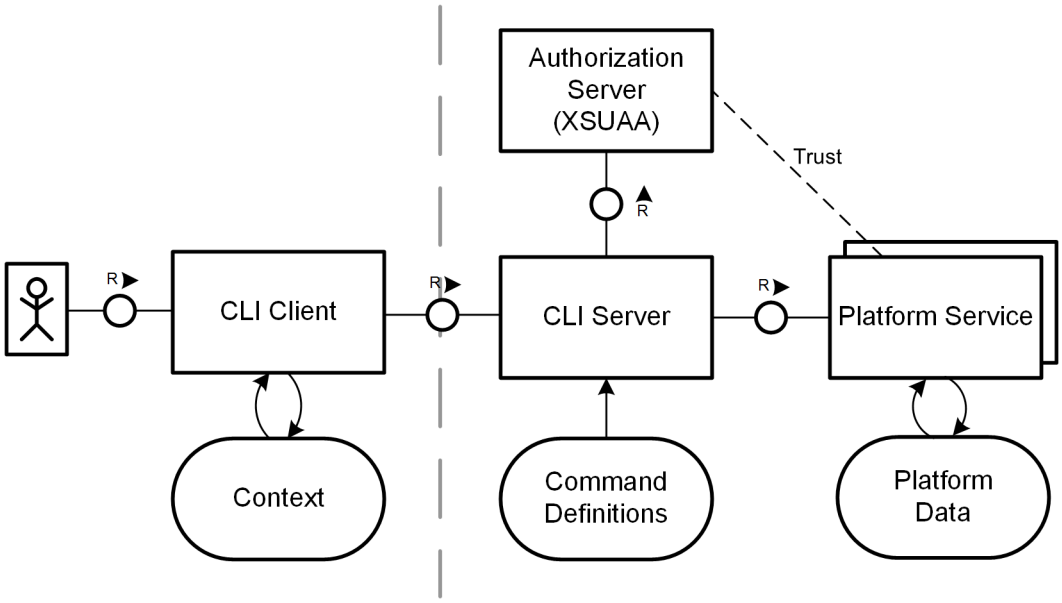
- Download and Start Using the btp CLI Client [page 1319]
- Set the Default Command Context [page 1338]

Apart from reading the documentation, you can also check out the Get Started with btp CLI tutorial.

Learn about frequent administrative tasks you can perform using the btp CLI:

- Working with Global Accounts, Directories, and Subaccounts Using the btp CLI [page 1343]
- Setting Entitlements Using the btp CLI [page 1345]

How the btp CLI Works



You download the btp CLI client to your local desktop and access it through the shell of your operating system. The client then accesses all required platform services through its backend, the CLI server, where the command definitions are stored. The CLI server delegates authentication and authorization to the authorization server, and forwards trust to the platform services, which then take care of authorization at the execution of each command.

Related Information

Download and Start Using the btp CLI Client [page 1319]

Command Syntax of the btp CLI [page 1323]
Log in [page 1330]
Set the Default Command Context [page 1338]
Commands in the btp CLI [page 1343]
Cloud Management Tools — Feature Set Overview

6.2.1 Download and Start Using the btp CLI Client

To use the SAP BTP command line interface (btp CLI), you need to download the client first.

Context

The client is available for 64-bit versions of the following operating systems:

- Microsoft Windows
- Apple macOS
- Linux

> **i** Note
>
> With the release of version 2.0.0 on March 25, 2021, the executable file of the CLI client was renamed from **sapcp** to **btp**. All commands remain compatible, and we will support the sapcp CLI until September 2021. However, this documentation and all examples inside refer to the **btp CLI**, and we recommend to download the latest btp CLI client from the SAP Development Tools page. See Migrating from sapcp to btp [page 1322].

Procedure

1. Download the appropriate client for your operating system from SAP Development Tools or use the direct links to the latest version below. They are tar.gz archives that contain one executable file.
   - Latest version of CLI client for Microsoft Windows
   - Latest version of CLI client for Apple macOS
   - Latest version of CLI client for Linux

   > **i** Note
   >
   > You can also find older versions of the client at: https://tools.hana.ondemand.com/#cloud-btpcli. In exceptional cases, when your client is too new for the configured server, an error message tells you which version you need and where to find it.
2. Extract the client file (for example: `btp.exe`) to your local system and open a terminal session in the target folder.

On Windows, you can use powershell or an external program, such as WinRar, to extract the tar.gz file. Once you've unpacked the executable, you can enter `cmd` in the address bar of the folder. This opens the command prompt in this folder.

On macOS, you can open the tar.gz file by double-clicking in. Make sure that the executable is in your PATH, and open a terminal session.

On Linux, use the terminal to open the tar.gz file with `tar -xzf btp-cli-linux-amd64-latest.tar.gz` and make sure the executable is in your PATH.

3. Run `btp`. Note that you need read and write permissions in the target folder to run this executable.

> **i Note**
>
> On macOS, opening btp CLI may be blocked because it is from "an unidentifed developer". Please refer to the macOS documentation to learn how to bypass this.

You get a short welcome message with some useful information, such as the version of your client, how to display help, and how to log in.

The CLI creates a configuration file (`config.json`) in your user data directory.

4. Log in to your global account with `btp login`, which is interactive and will prompt for all required login information. Note that you need to provide the **subdomain of the global account** to log in. You can find this in the global account view in the cockpit. If you have more than one global account, see the *Switch Global Account* dialog. For details, see Log in [page 1330].
5. Once you're logged in, familiarize yourself with the btp CLI, for example with General Commands and Options in the btp CLI [page 1327], Command Syntax of the btp CLI [page 1323], or simply by trying out a few commands, such as the following:

```
btp list accounts/subaccount
```

```
btp list security/user
```

```
btp get security/user "name@example.com"
```

```
btp get accounts/global-account"
```

> **→ Tip**
>
> You can use the command autocompletion feature in the btp CLI to save keystrokes when entering command actions, group-object combinations, and their parameters in the command line. For more information, see Enable Command Autocompletion [page 1335].

6. If you're going to work in a subaccount of this global account, consider setting the default context to this subaccount using `btp target --subaccount <ID>`. See Set the Default Command Context [page 1338]. Tip: Use `btp list accounts/subaccount` to see the subaccounts of the global account.
7. To find out the current context and version, use `btp`.

Related Information

Set the Default Command Context [page 1338]
Command Syntax of the btp CLI [page 1323]
Get Help [page 1328]
Commands in the btp CLI [page 1343]
General Commands and Options in the btp CLI [page 1327]

6.2.1.1 Get Updates

We recommend updating the CLI client regularly to ensure that you can always use all new features.

Context

To find out the version of the CLI client you are using, run the command `btp --info` or simply `btp`.

Procedure

1. Get the latest version using the download links at: https://tools.hana.ondemand.com/#cloud-cpcli.
2. Extract the client file (for example: `btp.exe`) and replace the old file with this new one.
3. Work with the btp CLI as usual and enjoy the new features.

Related Information

Log in [page 1330]
Commands in the btp CLI [page 1343]

6.2.1.2 Migrating from sapcp to btp

The executable file was renamed from sapcp to btp, which makes it necessary to update all scripts to call btp instead of sapcp.

Context

From version 1.32.0 to version 2.0.0, the executable file was renamed from **sapcp** to **btp**. All commands remain compatible, but all scripts must be updated to call btp instead of sapcp. A new configuration file is created for you automatically.

The sapcp client version 1.32.0 will be supported until September 2021.

Procedure

1. Download the latest btp CLI client for your operating system from SAP Development Tools.
2. Update your scripts to call btp.
3. Open `btp` in a terminal session.

   Your sapcp configration file (`config.json`) is carried over into a new btp configuration folder in your user data directory:
   - Microsoft Windows: `C:\Users\<username>\AppData\Local\SAP\btp`
   - Apple macOS / Linux: `$HOME/.btp`

   The old sapcp configuration folder in the same directory is kept, so you can still use the sapcp client. If you no longer need it, you can delete the folder.
4. If you want to use command autocompletion, you can enable it with `btp enable autocomplete <SHELL>`.

Related Information

Specify the Location of the Configuration File [page 1341]
Enable Command Autocompletion [page 1335]
Command Syntax of the btp CLI [page 1323]

6.2.2 Command Syntax of the btp CLI

Each command consists of the base call `btp` followed by a verb (the action), a combination of group and object, and parameters.

> **i Note**
>
> With the release of version 2.0.0 on March 25, 2021, the executable file of the CLI client was renamed from **sapcp** to **btp**. All commands remain compatible, and we will support the sapcp CLI until September 2021. However, this documentation and all examples inside refer to the **btp CLI**, and we recommend to download the latest btp CLI client from the SAP Development Tools page. See Migrating from sapcp to btp [page 1322].

The btp CLI uses the following syntax:

```
btp [OPTIONS] ACTION GROUP/OBJECT [PARAMS]
```

The commands are ordered in groups. Words in caps are placeholders, and brackets [ ] denote optionality. Here's is one example with the help option and no parameters before we outline the entire syntax:

> **Sample Code**
>
> ```
> btp --help list accounts/subaccount
> ```

- `btp` is the base call to start each command
- OPTIONS can be added to each command, for example to get help `--help` or execute a command in verbose mode `--verbose`. Note that the `--help` option can also be placed at the end of a command.
- ACTION is the verb. Depending on the GROUP/OBJECT combination, different verbs are available, such as `get`, `list`,`create`, `delete`, `assign`, `unassign`, `add`, `remove`. For a complete list, use `btp --help`.
- The GROUP/OBJECT combination specifies the entity that the action is carried out on. For example, all commands related to users and their authorizations belong to the **security** group, in which **objects** such as role, role-collection, and user are available.
- PARAMETERS are passed with most commands. Some commands have one positional parameter, which is entered directly after the command. All further parameters have a key and can be optional. The command help specifies the optional parameters as such. Ffor example, in `btp assign security/role-collection "Global Account Administrator" --to-user example@mail.com --of-idp my-idp`, "Global Account Administrator" is the positional parameter, and the other two parameters have keys.

> **i Note**
>
> The commands that you type into the command-line are interpreted and executed by the shell. Make sure you're familiar with your shell to avoid unexpected interferences. For examples of correct escaping, see Passing JSON Parameters in the Command Line [page 1325].

> **→ Tip**
>
> You can use the command autocompletion feature in the btp CLI to save keystrokes when entering commands actions, group-object combinations, and their parameters in the command line. For more information, see Enable Command Autocompletion [page 1335].

A Few Commands to Get Started

Here are a few commands for you to try out once you're logged in (Log in [page 1330]):

```
btp list accounts/subaccount
```

```
btp list security/user
```

```
btp get security/user "name@example.com"
```

```
btp list accounts/subscription
```

Example

In this example, we assign the **Global Account Administrator** role collection to user **name@example.com** and try out some options.

Using the `--help` option to display command-specific help to learn how to use the command:

```
btp --help assign security/role-collection
```

```
btp assign security/role-collection --help
```

Command with positional parameter and one mandatory parameter:

```
btp assign security/role-collection "Global Account Administrator" --to-user
"name@example.com"
```

Command with positional parameter, mandatory parameter, and optional parameter:

```
btp assign security/role-collection "Global Account Administrator" --to-user
"name@example.com" --of-idp "my-idp"
```

The same command in verbose mode:

```
btp --verbose assign security/role-collection "Global Account Administrator" --
to-user "name@example.com" --of-idp "my-idp"
```

Related Information

Set the Default Command Context [page 1338]

6.2.2.1 Passing JSON Parameters in the Command Line

Depending on the shell you use, the escaping rules are different. Here you'll find examples of correct escaping and quotes when passing JSON objects in the command line using different shells.

The table below gives an overview of the escaping rules in these commonly used shells.

Shell	Correct Escaping and Quotes
Bash	Use `--parameter "VALUE"` and escape quotes with `\"` within `VALUE` Use `--parameter 'VALUE'` and do not escape quotes within `VALUE`
Windows Command Prompt	Use `--param "VALUE"` and escape quotes with `\"` within `VALUE`
Windows PowerShell	Use `--param 'VALUE'` and escape quotes with `\'` within `VALUE`

> → Tip
>
> The CLI client provides examples in the command help for all commands. The examples that include JSON parameters use formatting that is compliant with Bash (Unix/Linux operating system, macOS) and Windows Command Prompt.

The command-line examples below are based on the following JSON:

```
[
  {
    "key": "Key 1",
    "value": "Value 1"
  },
  {
    "key": "Key 2",
    "value": "Value 2"
  }
]
```

Bash (Linux, macOS)

Option 1

```
--parameter '[{"key": "Key 1","value": "Value 1"},{"key": "Key 2","value":
"Value 2"}]'
```

For example, when creating a subaccount (`btp create accounts/subaccount`) with custom properties `Landscape` = Dev and `Department` = HR, use the following syntax:

Sample Code

```
--custom-properties '[{"key": "Landscape","value": "Dev"},{"key":
"Department","value": "HR"}]'
```

Option 2

```
--parameter "[{\"key\":\"Key 1\",\"value\":\"Value 1\"},{\"key\":\"Key 2\",
\"value\":\"Value 2\"}]"
```

For example, when creating a subaccount (`btp create accounts/subaccount`) with custom properties `Landscape` = Dev and `Department` = HR, use the following syntax:

Sample Code

```
--custom-properties "[{\"key\":\"Landscape\",\"value\":\"Dev\"},{\"key\":
\"Department\",\"value\":\"HR\"}]"
```

Windows Command Prompt

```
--parameter "[{\"key\":\"Key 1\",\"value\":\"Value 1\"},{\"key\":\"Key 2\",
\"value\":\"Value 2\"}]"
```

For example, when creating a subaccount (`btp create accounts/subaccount`) with custom properties `Landscape` = Dev and `Department` = HR, use the following syntax:

Sample Code

```
--custom-properties "[{\"key\":\"Landscape\",\"value\":\"Dev\"},{\"key\":
\"Department\",\"value\":\"HR\"}]"
```

Windows PowerShell

```
--parameter '[{\"key\": \"Key 1\",\"value\": \"Value 1\"},{\"key\": \"Key 2\",
\"value\": \"Value 2\"}]'
```

For example, when creating a subaccount (`btp create accounts/subaccount`) with custom properties `Landscape` = Dev and `Department` = HR, use the following syntax:

Sample Code

```
--custom-properties '[{\"key\": \"Landscape\",\"value\": \"Dev\"},{\"key\":
\"Department\",\"value\": \"HR\"},{\"key\": \"Flagged\",\"value\": \"\"}]'
```

Related Information

Command Syntax of the btp CLI [page 1323]
Working with Global Accounts, Directories, and Subaccounts Using the btp CLI [page 1343]
Working with External Resource Providers Using the btp CLI
Working with Environments Using the btp CLI

6.2.3 General Commands and Options in the btp CLI

Learn how to work with the SAP BTP command line interface (btp CLI). For example, how to log in, get help, and set a default context for commands.

General Commands

```
btp login
```

```
btp logout
```

```
btp target
```

General Options for Each Command

The following options are available for each command. They need to be typed right after the base call `btp`, and they can be combined (for example, `btp --verbose --help list accounts/subaccount`. The `--help` option also works at the end of a command call.

Options	
--config	Specifies the location of the configuration file. See Specify the Location of the Configuration File [page 1341].
--info	Displays version and current context. Note that this option only works on its own (`btp --info`) and cannot be added to other command calls. You can also just use `btp` to display the info. See View Version and Current Context [page 1329].
--help	Displays help. See Get Help [page 1328].
--verbose	Prints tracing information for support. See Troubleshooting and Support.
--format	Changes the output format of a command to JSON. See Change the Output Format to JSON [page 1339].

Get Help [page 1328]

Get help in the btp CLI with the `--help` option.

View Version and Current Context [page 1329]
: To find out the current context you're working in, run the command `btp --info` or simply `btp`.

Log in [page 1330]
: Log in with the btp CLI is on global account level.

Log out [page 1334]
: Logging out of the configured server removes all user-specific data from the configuration file.

Enable Command Autocompletion [page 1335]
: Use command autocompletion to save keystrokes when entering command actions, group-object combinations, and their parameters in the SAP BTP command line interface (btp CLI).

Set the Default Command Context [page 1338]
: Change the default context for all command calls to the global account, a directory, or a subaccount by using the `btp target` command.

Change the Output Format to JSON [page 1339]
: Use the `--format json` option to change the output format of a command to JSON.

Specify the Location of the Configuration File [page 1341]
: You can change the location of the configuration file by using the `--config` option.

Related Information

Commands in the btp CLI [page 1343]

6.2.3.1 Get Help

Get help in the btp CLI with the `--help` option.

- To display general help, use `btp --help`. This lists all commands, ordered in group/object combinations, with the possible actions.
- To display help for a specific command, type `--help` at the beginning or end of the the respective command. This displays information about the usage, its parameters, a description, helpful tips, and examples.

Example

Sample Code

```
btp --help list accounts/subaccount
```

Sample Code

```
btp list accounts/subaccount --help
```

```
>btp list accounts/subaccount --help
Usage: btp [OPTIONS] list accounts/subaccount --global-account SUBDOMAIN

List all the subaccounts in a global account, including the subaccounts in directories.

Parameter:
  --global-account,-ga SUBDOMAIN  The subdomain of the global account.

Example:
  btp list accounts/subaccount

Tips:
  You must be assigned to the global account admin or viewer role.
  To view all directories and subaccounts in a global account, use 'btp get accounts/global-account --show-hierarchy'.
  To view the subaccounts in a specific directory, use 'btp get accounts/directory'.
  For details about the directory features, see command help for 'btp create accounts/directory'.

See also:
  btp get accounts/subaccount
  btp get accounts/global-account
  btp get accounts/directory

OK
```

Parent topic: General Commands and Options in the btp CLI [page 1327]

Related Information

View Version and Current Context [page 1329]
Log in [page 1330]
Log out [page 1334]
Enable Command Autocompletion [page 1335]
Set the Default Command Context [page 1338]
Change the Output Format to JSON [page 1339]
Specify the Location of the Configuration File [page 1341]

6.2.3.2 View Version and Current Context

To find out the current context you're working in, run the command `btp --info` or simply `btp`.

Procedure

Use `btp --info` or just `btp`.

The client displays its own version, usage information, the CLI server URL, the global account, directory, and subaccount you're working in and their hierarchy, as well as the location of the configuration file.

Task overview: General Commands and Options in the btp CLI [page 1327]

Related Information

Get Help [page 1328]
Log in [page 1330]
Log out [page 1334]
Enable Command Autocompletion [page 1335]
Set the Default Command Context [page 1338]
Change the Output Format to JSON [page 1339]
Specify the Location of the Configuration File [page 1341]
Set the Default Command Context [page 1338]

6.2.3.3 Log in

Log in with the btp CLI is on global account level.

Prerequisites

- Your global account is on feature set B. See Cloud Management Tools — Feature Set Overview.
- To log in to your global account, you need the subdomain of your global account, which you find in the cockpit in the global account view or under *Switch Global Account*.
- If your operator has provided you with a CLI server URL, you'll need to enter it at login. If not, you can use the one proposed by the btp CLI at login.

Context

When you log in to your global account with the btp CLI, a token is created and stored on your computer that allows to close and reopen the command line without losing your login. With each command call, this token is renewed and valid for 24 hours. So, if you take a longer break from working with the btp CLI, you'll have to log in again. If you want to end your login earlier, you can use `btp logout`.

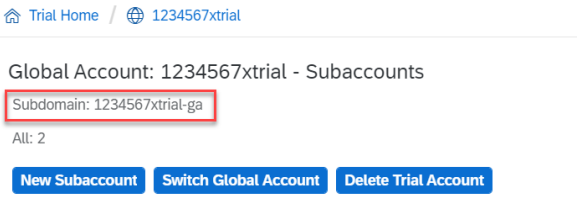
Alternatively, you can also log in with single sign-on directly at your identity provider through a web browser. See Log In Through a Browser [page 1333].

Procedure

Use `btp login`. The btp CLI prompts for all login information, but optionally, you can provide the required information as parameters.

Usage: `btp [OPTIONS] login [PARAMS]`

Parameters

`--url <URL>`	The client proposes this URL: `https://cpcli.cf.eu10.hana.ondemand.com`, which you can confirm by pressing `ENTER`. If your operator has provided you with a different server URL, you can specify it here. Note that when you enter a new server URL for the first time, you're asked to confirm that you trust it.
`--subdomain <GLOBALACCOUNT>`	The subdomain of the global account you want to log in to. You should have obtained the subdomain from your operator; but you can also find it in the cockpit in the global account view or under *Switch Global Account* dialog.  **i Note** If you don't find the subdomain of the global account in your cockpit, ensure that you're using a global account on SAP BTP feature set B. See Cloud Management Tools — Feature Set Overview.
`--user <USER>`	Your user name, usually an email address.
`--password <PASSWORD>`	Your password. Note that if two-factor authentication is enabled, concatenate your password and the passcode in a single string. **→ Tip** We don't recommend to provide the password with this parameter, as it appears in plain text and may be recorded in your shell history. Rather, enter it when you're prompted.

```
btp login --url https://cpcli.cf.eu10.hana.ondemand.com --subdomain my-global-account --user name@example.com
```

> **i Note**
>
> To log on with SAP Universal ID, you need a browser in the logon process. Use the option `--sso`.
>
> For more information, see Log In Through a Browser [page 1333].
>
> Otherwise log on with the password associated with your account (S-user or P-user) in the default identity provider, SAP ID service.
>
> If you've forgotten this password and this user is associated with your SAP Universal ID user, reset your password.
>
> For more information, see SAP Note 3085908.

If you've logged in before, the server URL, the subdomain, and the user from the last login are suggested. You can then press `Enter` to confirm, or type in different values.

```
CLI server URL [https://cpcli.cf.eu10.hana.ondemand.com]>
Subdomain [my-global-account]>
User [name@example.com]>
```

Results

Upon successful login, the btp CLI creates a folder (`btp`) and a configuration file (`config.json`) in the default location of your user data directory:

- Microsoft Windows: `C:\Users\<username>\AppData\Local\SAP\btp\config.json`
- Apple macOS / Linux: `$HOME/.btp/config.json`
  To change this location, use the `SAPCP_CLIENTCONFIG` environment variable or the `--config` option, which you can append to each command call. See Specify the Location of the Configuration File [page 1341].

> **→ Tip**
>
> You've logged in to the global account and all commands are executed in the context of this global account. To change this default context for subsequent commands to a subaccount or directory of this global account, use `btp target`. See Set the Default Command Context [page 1338].

Task overview: General Commands and Options in the btp CLI [page 1327]

Related Information

Get Help [page 1328]
View Version and Current Context [page 1329]
Log out [page 1334]
Enable Command Autocompletion [page 1335]

Set the Default Command Context [page 1338]
Change the Output Format to JSON [page 1339]
Specify the Location of the Configuration File [page 1341]
Commands in the btp CLI [page 1343]

6.2.3.3.1 Log In Through a Browser

Use the single sign-on flag (`btp login --sso`) to log in to your identity provider through a browser instead of passing username and password on the command line.

Prerequisites

Login through a browser is only available if you use the default server URL that is proposed by the CLI during login (`https://cpcli.cf.eu10.hana.ondemand.com`). This is the case unless your operator has provided you specifically with a different server URL.

Procedure

1. Enter the following and press `ENTER` to be prompted for the server URL and the subdomain of your global account:

   ```
   btp login --sso
   ```

   Alternatively, you can pass the server URL and the subdomain of your global account as parameters:

   ```
   btp login --sso --url https://cpcli.cf.eu10.hana.ondemand.com --subdomain <my-subdomain>
   ```

   To suppress the automatic browser-opening, you can run the command as follows:

   ```
   btp login --sso manual
   ```

2. Authenticate directly at your identity provider through your browser.

   If you already have an active session, the identity provider immediately transfers a token to the client, which completes your login. Otherwise, you need to enter your credentials.

   You've successfully logged in. You can return to the command line to work in your global account.

Related Information

Log in [page 1330]

Commands in the btp CLI [page 1343]
General Commands and Options in the btp CLI [page 1327]

6.2.3.4 Log out

Logging out of the configured server removes all user-specific data from the configuration file.

Context

Once you're finished using the btp CLI and you want to ensure that your locally stored credentials are immediately deleted, you can run the logout command. If you choose not to log out, your credentials will expire 24 hours after you last command execution, but the next time you log in, the btp CLI will propose your current subdomain and user so you won't have to type it in again.

Procedure

To log out, use `btp logout`.

This terminates your active logout session and ensures that all user-specific data is removed. The next time you log in, you will have to type in the subdomain of the global account and your user.

Task overview: General Commands and Options in the btp CLI [page 1327]

Related Information

Get Help [page 1328]
View Version and Current Context [page 1329]
Log in [page 1330]
Enable Command Autocompletion [page 1335]
Set the Default Command Context [page 1338]
Change the Output Format to JSON [page 1339]
Specify the Location of the Configuration File [page 1341]
Log in [page 1330]

6.2.3.5 Enable Command Autocompletion

Use command autocompletion to save keystrokes when entering command actions, group-object combinations, and their parameters in the SAP BTP command line interface (btp CLI).

Context

Autocompletion in btp CLI currently supports the following shells:

- Bash
- PowerShell
- Zsh

> **i** Note
>
> The respective shell must be installed on your operating system before enabling autocomplete.

Once autocomplete is enabled (it's disabled by default), you use the autocomplete feature as follows in the command line:

- Enter a partial command action, group-object combination, or parameter, and then press the `Tab` key. The command line either automatically completes your command or, when there's more than one option available, it displays a list of suggested command actions/options/parameters.
- When a suggestion list is displayed, depending on your shell, use the `Tab` or arrow keys to move through the list and press `Enter` to make a selection.

Examples

The following examples show various ways that you can use autocompletion:

- Enter `btp` and press `Tab` to display all available actions:

```
./btp TAB
add           create       enable       list         logout       register
subscribe     unassign     unsubscribe
assign        delete       get          login        move         remove
target        unregister   update
```

- Partially enter `btp cre` and press `Tab` to autocomplete the command to `btp create`. Then, press `Tab` again to display a suggested list of group-object combinations:

```
./btp create TAB
accounts/directory             accounts/resource-provider     security/
role                  services/binding
accounts/environment-instance  accounts/subaccount            security/role-
collection       services/instance
```

- Partially enter a group and press `Tab` to display a suggested list of objects:

```
./btp create accountsTAB
accounts/directory             accounts/environment-instance  accounts/
resource-provider     accounts/subaccount
```

- Partially enter a parameter and press `Tab` to display a suggested list of parameters:

```
./btp create accounts/subaccount -TAB
--beta-enabled
--custom-properties
--description
--directory
--display-name
--global-account
--region
--subaccount-admins
--subdomain
--used-for-production
```

Procedure

1. Use `btp enable autocomplete <SHELL>` to enable command autocompletion for a specified shell.

   The valid values for the supported shells are:

   - `bash`
   - `powershell`
   - `zsh`

   > ### Sample Code
   >
   > ```
   > btp enable autocomplete zsh
   > ```

2. The client asks for confirmation to install the autocomplete plugin script at the specified location. Enter "y" or "yes" to continue.

   Apart from the script being added to your file system, the RC file of your shell is modified to call this script at startup. The client looks for this RC file in your system and proposes possible files. You can choose to accept a proposal or specify a different RC file. If no RC file is found, the clients prints an error message with the full path of the missing file, so you can create it and run the enable command again.

3. Enter the option of your choice.

   To specify a custom path, first choose "Custom" and then enter the full path of the RC file to use.

   For example:

   ```
   ./btp enable autocomplete powershell
   This will install the autocomplete plugin script for powershell to C:\AppData
   \btp\autocomplete\scripts. Do you want to continue? [no]>y
   Which RCFile should be used for the installation?
   1: C:\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
   2: Custom
   Enter option>2
   Enter the full path of your RCFile>C:\Documents\WindowsPowerShell
   \Microsoft.PowerShell_my-custom-profile.ps1
   ```

4. Start a new terminal session to activate the installed autocomplete script.

Results

When you enable command autocompletion, a script containing all the autocomplete commands is downloaded and installed in your file system.

The autocompletion option remains enabled in future sessions in your current client, until you disable it. To disable command autocompletion and uninstall the autocomplete script, run the following command:

```
btp disable autocomplete <SHELL>
```

You can run either `btp` or `btp --info` to see if command autocompletion is currently enabled and where the autocomplete script for your shell is located. If you don't see a line specifying the location of the autocomplete script, then it's disabled.

> → Tip
>
> If you see a discrepancy between the version of the autocomplete script and the client, the update of the autocomplete script might have failed. In such a case, try to disable and enable the autocomplete feature again.

Whenever you start a new btp CLI terminal session, the installed autocomplete scripts are automatically updated to include the latest commands. If a script is updated, you're prompted to restart your terminal session to load the newest autocomplete information.

If disabling the command autocompletion fails or you have uninstalled the btp CLI client without disabling autocompletion, you can manually remove traces of the autocomplete installation in your shell initialization file (RC or profile file depending on your shell) by deleting the line that starts with `SAPCP_CLI_AUTOCOMPLETE`.

Task overview: General Commands and Options in the btp CLI [page 1327]

Related Information

Get Help [page 1328]
View Version and Current Context [page 1329]
Log in [page 1330]
Log out [page 1334]
Set the Default Command Context [page 1338]
Change the Output Format to JSON [page 1339]
Specify the Location of the Configuration File [page 1341]

6.2.3.6 Set the Default Command Context

Change the default context for all command calls to the global account, a directory, or a subaccount by using the `btp target` command.

Context

Targeting works along the hierarchy of your account model:

- After login, the global account is targeted by default.
- Commands that only work on a directory or global account level will be executed in the parent directory or global account of the current target.
- When targeting a subaccount or directory, you can execute commands in its parent directory or global account by using '-dir' or '-ga' without a value.

Procedure

Use `btp target [PARAMS]` to set the context for subsequent commands.

Usage: `btp [OPTIONS] target [PARAMS]`

Parameters

`--global-account, -ga <SUBDOMAIN>`	Only the global account of the active login can be targeted. As only one active login is possible, SUBDOMAIN can be omitted.
`--directory, -dir<ID>`	The ID of the directory to be targeted. You can find the directory ID by running `btp get accounts/global-account --show-hierarchy`
`--subaccount, -sa <ID>`	The ID of the subaccount to be targeted. You can find this ID by running `btp list accounts/subaccount`.

Results

Once you have set the context to a subaccount, all subsequent commands are executed there, unless you specify a different one by providing one of the parameters directly in a command call.

> → Tip
>
> To find out your current context, use `btp --info` or simply `btp`.

Example

To set the context back to the global account, use `btp target -ga`.

Task overview: General Commands and Options in the btp CLI [page 1327]

Related Information

Get Help [page 1328]
View Version and Current Context [page 1329]
Log in [page 1330]
Log out [page 1334]
Enable Command Autocompletion [page 1335]
Change the Output Format to JSON [page 1339]
Specify the Location of the Configuration File [page 1341]
Commands in the btp CLI [page 1343]
Global Accounts

6.2.3.7 Change the Output Format to JSON

Use the `--format json` option to change the output format of a command to JSON.

Context

The standard output format of the btp CLI is text, formatted in a way that an interactive user of the btp CLI can easily read it. You can switch this to JSON output to make use of automation. Also, The JSON output often cludes more information than the respective text output, as the text output is optimized for human readability and therefore limited to the most relevant information.

Procedure

Start commands with `--format json`. Currently, the only valid value is `json`.

```
btp --format json list accounts/subaccount
```

Results

Example with text output

```
btp list accounts/subaccount
subaccounts in global account c8XXXX3d-6XX7-4XXb-8dXX-89XXXX372eXX...
subaccount id:                          display
name:                               subdomain:
region:        beta-enabled:   parent id:
state:            state
message:

88XXXX80-9844-4XX7-8XXd-beXXXX42bfb2   my-
subaccount-1                            my-CF-subdomain1
us10-TEST      false           c8XXXX3d-6XX7-4XXb-8dXX-89XXXX372eXX
OK                Subaccount
created.

ebXXXX3e-f3a2-4XXf-a080-3eXXXX9b9ced   my-
subaccount-2                            my-CF-subdomain2
us10-TEST      false           c8XXXX3d-6XX7-4XXb-8dXX-89XXXX372eXX
OK                Subaccount created.
```

Example with JSON output

```
btp list --format json accounts/subaccount
{
  "value": [
    {
      "guid": "88XXXX80-9844-4XX7-8XXd-beXXXX42bfb2",
      "displayName": "my-subaccount-1",
      "globalAccountGUID": "c8XXXX3d-6XX7-4XXb-8dXX-89XXXX372eXX",
      "parentGUID": "c8XXXX3d-6XX7-4XXb-8dXX-89XXXX372eXX",
      "parentType": "ROOT",
      "region": "us10-TEST",
      "subdomain": "my-CF-subdomain1",
      "betaEnabled": false,
      "usedForProduction": "UNSET",
      "description": null,
      "expiryDate": null,
      "state": "OK",
      "stateMessage": "Subaccount created.",
      "createdDate": 1602759737909,
      "createdBy": "name@email.com",
      "modifiedDate": 1602759737909,
      "zoneId": null
    },
    {
      "guid": "ebXXXX3e-f3a2-4XXf-a080-3eXXXX9b9ced",
      "displayName": "CF TEST",
      "globalAccountGUID": "c8XXXX3d-6XX7-4XXb-8dXX-89XXXX372eXX",
      "parentGUID": "c8XXXX3d-6XX7-4XXb-8dXX-89XXXX372eXX",
      "parentType": "ROOT",
      "region": "us10-TEST",
      "subdomain": "my-CF-subdomain2",
      "betaEnabled": false,
      "usedForProduction": "UNSET",
      "description": null,
      "expiryDate": null,
      "state": "OK",
      "stateMessage": "Subaccount created.",
      "createdDate": 1602760244477,
      "createdBy": "name@email.com",
      "modifiedDate": 1602760244477,
      "zoneId": null
```

```
      }
    ]
}
```

Task overview: General Commands and Options in the btp CLI [page 1327]

Related Information

Get Help [page 1328]
View Version and Current Context [page 1329]
Log in [page 1330]
Log out [page 1334]
Enable Command Autocompletion [page 1335]
Set the Default Command Context [page 1338]
Specify the Location of the Configuration File [page 1341]

6.2.3.8 Specify the Location of the Configuration File

You can change the location of the configuration file by using the `--config` option.

Context

Upon successful login, the btp CLI creates a folder (`btp`) and a configuration file (`config.json`) in the default location of your user data directory:

- Microsoft Windows: `C:\Users\<username>\AppData\Local\SAP\btp\config.json`
- Apple macOS / Linux: `$HOME/.btp/config.json`

This folder serves as the working directory of the btp CLI, that is, it's necessary for its proper functioning, and is used with each command execution.

If you want the configuration file to be created in a different folder, you can use the `--config` option in your login command and then specify this location in each command call with this `--config` option.

Procedure

1. Specify the location of the configuration file with your login command:

```
btp --config "<file path>" login
```

2. Specify this location either with the `SAPCP_CLIENTCONFIG` environment variable, or use the `--config` option with each subsequent command call.

> Sample Code
>
> ```
> btp --config "<file path>"
> ```

Example

Let's assume you want to work in two subaccounts in parallel, using two terminals. For example, with the first terminal (A), you want to work in a subaccount with ID 1000, with the second terminal (B) you want to work in a subaccount with ID 2000, and you want to list the users in each subaccount.

Terminal A	Terminal B
1. Log in to your global account using the default location of the configuration file: `btp login` Run all commands as usual. The btp CLI uses the default configuration file.	1. Log in to your global account using a different location of the configuration file: `btp --config "C:\my-cli-folder" login` Use this option with each command call.
2. Set the default context to subaccount 1000: `btp target --subaccount 1000`	2. Set the default context to subaccount 2000: `btp --config "C:\my-cli-folder" target --subaccount 2000`
3. List the users of subaccount 1000: `btp list security/user`	3. List the users of subaccount 2000: `btp --config "C:\my-cli-folder" list security/user`

Task overview: General Commands and Options in the btp CLI [page 1327]

Related Information

Get Help [page 1328]
View Version and Current Context [page 1329]
Log in [page 1330]
Log out [page 1334]
Enable Command Autocompletion [page 1335]
Set the Default Command Context [page 1338]
Change the Output Format to JSON [page 1339]

6.2.4 Commands in the btp CLI

A list of the tasks and respective commands that are available in the SAP BTP command line interface (btp CLI) when working with global accounts and directories that include subacccounts in the Neo Environment.

You can create, update, and move Neo subaccounts to directories using the btp CLI, but if you want to actually access a subaccount in the Neo environment, you need to work with the Neo command console or the SAP BTP cockpit. See Working with Neo subaccounts using the SAP BTP command line interface [page 820] for the known conditions and scope when working with Neo subaccounts in the btp CLI.

> → Tip
>
> You can find extensive help about each command directly in the btp CLI by using the `--help` option. For example, use one of the following commands for help on `btp list accounts/subaccount`:
>
> ```
> btp --help list accounts/subaccount
> ```
>
> ```
> btp list accounts/subaccount --help
> ```

Working with Global Accounts, Directories, and Subaccounts Using the btp CLI [page 1343]
Use the SAP BTP command line interface (btp CLI) to manage operations with global accounts, directories, and subaccounts.

Setting Entitlements Using the btp CLI [page 1345]
Use the SAP BTP command line interface (btp CLI) to set entitlements to define the functionality or permissions available for users of global accounts, directories, and subaccounts.

Related Information

Log in [page 1330]
Command Syntax of the btp CLI [page 1323]
Get Help [page 1328]
Set the Default Command Context [page 1338]

6.2.4.1 Working with Global Accounts, Directories, and Subaccounts Using the btp CLI

Use the SAP BTP command line interface (btp CLI) to manage operations with global accounts, directories, and subaccounts.

Working with Global Accounts

Task	Run the command...
Get details about a global account, and the account structure (directories and subaccounts) of the global account.	`btp get accounts/global-account`
Update the display name and/or description of a global account.	`btp update accounts/global-account`

For more information, see Global Accounts.

Working with Directories

Directories allow you to organize and manage your subaccounts according to your technical and business needs.

Task	Run the command...
Get details about a directory and list the subaccounts and subdirectories in the directory	`btp get accounts/directory`
Create a directory	`btp create accounts/directory`
Update a directory	`btp update accounts/directory`
Delete a directory	`btp delete accounts/directory`
Change the set of enabled features (user and entitlement management) for a directory	`btp enable accounts/directory`
List the custom properties assigned to a directory	`btp list accounts/custom-property`

For more information, see Directories [Feature Set B] [page 12].

Working with Subaccounts

Task	Run the command...
List all subaccounts in a global account	`btp list accounts/subaccount`
Get details about a subaccount	`btp get accounts/subaccount`
Create a subaccount	`btp create accounts/subaccount`
Update a subaccount	`btp update accounts/subaccount`
Move a subaccount	`btp move accounts/subaccount`
List the custom properties assigned to a subaccount	`btp list accounts/custom-property`
Get all available regions for global account	`btp list accounts/available-region`

> **i Note**
>
> To delete a Neo subaccount, you need to use instead the Neo console or the cockpit.

For more information, see Subaccounts.

Parent topic: Commands in the btp CLI [page 1343]

Related Information

Setting Entitlements Using the btp CLI [page 1345]
Set the Default Command Context [page 1338]
Custom Properties [Feature Set B] [page 14]

6.2.4.2 Setting Entitlements Using the btp CLI

Use the SAP BTP command line interface (btp CLI) to set entitlements to define the functionality or permissions available for users of global accounts, directories, and subaccounts.

Task	Run the command...
Get all the entitlements and quota assignments for a global account, directories, and subaccounts	`btp list accounts/entitlement` **! Restriction** It is not possible to list the entitlements of a Neo subaccount with `btp list accounts/entitlements --subaccount <ID>`. You can, however, list the entitlements for a directory that includes Neo subaccounts with `btp list accounts/entitlements --directory <ID>`.
Assign or update an entitlement to a subaccount or a directory	`btp assign accounts/entitlement`

Parent topic: Commands in the btp CLI [page 1343]

Related Information

Working with Global Accounts, Directories, and Subaccounts Using the btp CLI [page 1343]

6.3 Account Administration in the Neo Console Client

Use the SAP BTP console client for the Neo environment for subaccount management in the Neo environment.

To learn more about	See
Downloading and setting up the console client	Set Up the Console Client
Opening the tool and working with the commands and parameters	Using the Console Client Console Client Video Tutorial
Groups of console client commands	Console Client Commands
Ranges and types of exit codes	Exit Codes
Machine-readable output of commands	Machine-Readable Command Output
Verbose mode of output	Verbose Mode of the Console Commands Output

6.3.1 Create Subaccounts Using the Console Client in the Neo Environment

You can create subaccounts using the console client in the Neo environment.

Prerequisites

- You have an enterprise account.
- You must be a member of the global account that contains the subaccount.
- You work in the Neo environment.
- You set up the console client. See Set Up the Console Client.

> → Recommendation
>
> Before creating your subaccounts, we recommend you learn more about Setting Up Your Account Model.

Context

Procedure

Open a command line and enter the following string:

```
neo create-account --display-name <subaccount_display_name> --account
<subaccount_technical_name> --user <e-mail_or_user> --host <host>
```

Example:

```
neo create-account --account mysubaccount --display-name mynewsubaccount --user
myuser --host hana.ondemand.com
```

> **i** Note
>
> For more information on creating new subaccounts and cloning existing subaccounts using the console client, see create-account [page 1379].

6.3.2 Add Quotas to Subaccounts Using the Console Client

You can use the Neo console client to add quotas to subaccounts

Prerequisites

- An enterprise account.
- You must be a member of the global account that contains the subaccount.
- Set up the console client. See Set Up the Console Client.

Context

Procedure

Open a command line and enter the following string:

```
neo set-quota --account <subaccount_technical_name> --host <host> --user <e-
mail_or_user> --amount <quota_type>:<integer_amount_of_quota>
```

Example:

```
neo set-quota --account mysubaccount --user myuser --host eu1.hana.ondemand.com
--amount lite:2
```

> **Note**
>
> For more information on adding quotas to subaccounts using the console client, see set-quota.

6.3.3 Console Client for the Neo Environment

The console client for the Neo environment enables development, deployment and configuration of an application outside the Eclipse IDE as well as continuous integration and automation tasks. The tool is part of the SDK for SAP BTP, Neo environment. You can find it in the *tools* folder of your SDK location.

> **Note**
>
> The console client is related only to the Neo environment. For the Cloud Foundry environment use the Cloud Foundry command line interface. See Download and Install the Cloud Foundry Command Line Interface.

To learn more about	See
Downloading and setting up the console client	Set Up the Console Client
Opening the tool and working with the commands and parameters	Using the Console Client Console Client Video Tutorial
Groups of console client commands	Console Client Commands
Ranges and types of exit codes	Exit Codes
Machine-readable output of commands	Machine-Readable Command Output
Verbose mode of output	Verbose Mode of the Console Commands Output

6.3.3.1 Using the Console Client

You execute a console client command by entering *neo <command name>* with the appropriate parameters. To list all parameters available for the respective command, execute *neo help <command name>*.

- Opening the Console Client [page 1349]
- Properties File [page 1349]
- Command Line [page 1349]
- Parameter Priority [page 1350]
- Parameter Values [page 1350]
- Proxy Settings [page 1351]
- Output Mode [page 1351]

You can define the parameters of the different commands either directly in *the command line*, or, in *a properties file*:

```
neo <command name> <mandatory parameters> [optional parameters]
```

```
neo <command name> <properties file location>
```

Opening the Console Client

The console client is part of the SDK for SAP BTP,Neo environment. You can find it in the tools folder of your SDK installation.

To start it, open the command prompt and change the current directory to the <SDK_installation_folder>\tools location, which contains the neo.bat and neo.sh files.

Command Line

You can deploy the same application as in the example above by executing the following command directly in the command line:

```
neo deploy --account <subaccount technical name> --application <application
name> --source samples/deploy_war/example.war --user <user name or email>
```

Properties File

Within the *tools* folder, a file **example_war.properties** can be found in the **samples/deploy_war folder**. In the file, enter your own user and subaccount name:

```
##############################################
    # General settings - relevant for all commands #
    ##############################################
    # Your subaccount technical name
    account=<your subaccount>
    # Application name
    application=<your application name>
    # User for login to hana.ondemand.com.
    user=<email or user name>
    # Host of the admin server. Optional. Defaults to hana.ondemand.com.
    host=hana.ondemand.com
    ###############################################################
    # Deployment descriptor settings - relevant only for deployment #
    ###############################################################
    # List of file system paths to *.war files and folders containing them
    source=samples/deploy_war/example.war
```

Then execute the deployment with the following line:

```
neo deploy samples/deploy_war/example_war.properties
```

Note that you can have more than one properties file. For example, you can have a different properties file for each application or user in your subaccount.

Parameter Priority

Argument values specified in the command line override the values specified in the properties file. For example, if you have specified **`account=a`** in the properties file and then enter **`account=b`** in the command line, the operation will take effect in **`account b`**.

Parameter Values

Since the client is executed in a console environment, not all characters can be used in arguments. There are special characters that should be quoted and escaped.

Consult your console/shell user guide on how to use special characters as command line arguments.

For example, to use argument with value **`abc&()[]{}^=;!'+,`~123`** on Windows 7, you should quote the value and escape the **`!`** character. Therefore you should use **`"abc&()[]{}^=;^!'+,`~123"`**.

User

You can use your e-mail, SAP ID or user name.

Password

Do not specify your password in the properties file or as a command line argument. Enter a password only when prompted by the console client.

For example, use:

```
neo deploy samples/deploy_war/example_war.properties
```

instead of

```
neo deploy --password <mypassword > samples/deploy_war/example_war.properties
```

> **! Restriction**
>
> Your password cannot start with the "@" character.

> **i Note**
>
> You can't log on with SAP Universal ID. Log on with the password associated with your account (S-user or P-user) in the default identity provider, SAP ID service.
>
> If you've forgotten this password and this user is associated with your SAP Universal ID user, reset your password.
>
> For more information, see SAP Note 3085908.

Proxy Settings

If you work in a proxy environment, before you execute commands, you need to configure the proxy.

For more information, see Set Up the Console Client.

Output Mode

You can configure the console to print detailed output during command execution.

For more information, see Verbose Mode of the Console Commands Output.

Related Information

Console Client Commands

6.3.3.2 Verbose Mode of the Console Commands Output

The console commands consist of:

- Local code - executed inside a local JVM, which is started when the command is started.
- Remote code - executed at back end (generally, the REST API that was called by the local code), which is started in a separate JVM on the cloud.

> **i Note**
>
> The trace level for remote code cannot be changed.

For local code execution, a LOG4J library is used. It is easy to be configured and, by default, there is a configuration file located inside the commands class path, that is `.../tools/lib/cmd`.

For each command execution, two appenders are defined - one for the session and one for the console. They both define different files for all messages that are logged by the SAP infrastructure and by `apache.http`. By default, the console commands output is written in a number of log files. However, you are allowed to change the `log4j.properties` file, and define additional appenders or change the existing ones. If you want, for example, the full output to be printed in the console (verbose mode), or you want to see details from the execution of specific libraries (partially verbose mode), you need to adjust the LOG4J configuration file.

For more information on how to configure the LOG4J, see https://logging.apache.org/ .

To adjust the level of a specific logger, you have to add `log4j.logger.<package> = <level>` in the code of the `log4j.properties` file.

For more information about the different levels, see https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/Level.html

In the file defined for the session, only loggers with level `ERROR` are logged. If you want, for example, to log debug information about the `apache.http` library, you have to change `log4j.category.org.apache.http=ERROR, session` to `log4j.category.org.apache.http=DEBUG, session`.

Example

This example demonstrates how you can change the output of command execution so that it is printed in the console instead of collecting the information within log files. To do this, open your SDK folder and go to directory `/tools/lib/cmd`. Then, open the `log4j.properties` file and replace its content with the code below.

> → Tip
>
> We recommend that you save the original content of the `log4j.properties` file. To switch back to the default settings, just revert the changes you did in the `log4j.properties` file.

```
##########
# Log levels
##########

log4j.rootLogger=INFO, console

log4j.additivity.rootLogger=false

log4j.category.com.sap = INFO, console
log4j.additivity.com.sap = false
log4j.category.org.apache.http = INFO, console
log4j.additivity.org.apache.http = false
log4j.category.org.apache.http.wire = INFO, console
log4j.additivity.org.apache.http.wire = false

##########
# System out console appender
##########

log4j.appender.console.Threshold=ALL

log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.Target=System.out

log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d %-5p [%t] %C: %m%n

log4j.appender.console.filter.1=org.apache.log4j.varia.StringMatchFilter
log4j.appender.console.filter.1.StringToMatch=>> Authorization: Basic
log4j.appender.console.filter.1.AcceptOnMatch=false
```

Related Information

Machine-Readable Command Output
Implementing Logging for Java Applications

6.3.3.3 Machine-Readable Command Output

Context

The console commands can return structured, machine-readable output. When you use the optional `--output` parameter in a command, the command returns values and objects in a format that a machine can easily parse. The currently supported output format is JSON.

Syntax: `--output <format>`

Accepted format value: **`json`**

Cases

- If a command supports structured output, it returns machine-readable result values.
- If a command does not (yet) support structured output,it returns basic information including the standard OUT/ERR output.
- If the command is invoked without the `--output` parameter, it works as before.

JSON Output Format

When `--output json` is specified, the console client prints out a single JSON object containing information about the command execution and the result, if available.

The JSON object contains the following properties:

Property Name	Type	Description
`command`	String	The name of the invoked `neo` command
`argLine`	String	The exact arguments line as specified by the calling script
`pid`	Name	The process ID of the invoked command
`exitCode`	Name	The process exit code of the invoked command (**0** = successful, everything else = failure)

Property Name	Type	Description
`errorMsg`	String	The message provided when the command implementation throws instance of `com.sap.jpaas.infrastructure.console.exception.CommandException`
`commandOutput`	String	The command output written to `system.out` and captured by the console client framework
`commandErrorOutput`	String	The command output written to `system.err` and captured by the console client framework
`result`	Object	The result object returned by the command following the new contract for structured, machine-readable output

Example

Here is a full example of a command (**`neo start`**) that supports structured output and displays result values:

```
{
    "command": "start",
    "argLine": "-a myaccount -b myapplication -h hana.ondemand.com -u myuser -p
******* -y",
    "pid": 6523,
    "exitCode": 0,
    "errorMsg": null,
    "commandOutput": "Requesting start for:
   application           : myapplication
   account               : myaccount
   host                  : https://hana.ondemand.com
   synchronous           : true
   SDK version           : 1.48.99
   user                  : myuser


[Tue Feb 25 18:07:19 CET 2014] Start request performed successfully.

Triggered start of application process.
Status: STARTING

[Tue Feb 25 18:07:19 CET 2014] Waiting for STARTED status.............
[Tue Feb 25 18:07:25 CET 2014] Status STARTING reached for 6161 ms
[Tue Feb 25 18:07:19 CET 2014] Waiting for STARTED
status..............................
[Tue Feb 25 18:08:47 CET 2014] Status STARTED reached for 87838 ms

web: STARTED

URL: https://myapplicationmyaccount.hana.ondemand.com

Access points:
  https://myapplicationmyaccount.hana.ondemand.com

Runtime: 1.47 (valid until 20-May-2015)
```

```
Application processes
ID          State       Last Change              Runtime
  fc735dc     STARTED     25-Feb-2014 18:07:48     1.47.10.2
",
    "commandErrorOutput": "",
    "result": {
        "status": "STARTED",
        "url": "https://myapplicationmyaccount.hana.ondemand.com",
        "accessPoints": [
            "https://myapplicationmyaccount.hana.ondemand.com",
            "https://myapplicationmyaccount.hana.ondemand.com/app2"
        ],
        "applicationProcesses": [
            {
                "id": "fc735dc",
                "state": "STARTED",
                "lastChange": "2014-02-25T18:07:48Z",
                "runtime": "1.47.10.2"
            }
        ]
    }
}
```

> **i Note**
>
> The shown command result is only an example and may look different in the real or future implementation. The output is similar for commands that do not support structured result values but the result property is then null.

Related Information

Exit Codes
Default Trace File

6.3.3.4 Console Client Commands

Prerequisites

To use the console client commands, you have to:

1. Download and install the SDK for SAP BTP,Neo environment and set up the console client. See Set Up the Console Client [page 847].
2. Open the console client. See Using the Console Client [page 1348].

Commands

> **i** Note
>
> You may need admin permissions in the cloud cockpit to be able to run some of these commands listed below.

Group	Commands
Local Server	install-local [page 1475]; deploy-local [page 1428]; start-local [page 1560]; stop-local [page 1564]
Deployment	deploy [page 1422]; start [page 1557]; status [page 1555] stop [page 1562]; restart [page 1528]; undeploy [page 1574] disable [page 1430]; enable [page 1443]; list-applications [page 1481]; list-runtimes [page 1500]; list-runtime-versions [page 1501]; display-application-properties [page 1432]; set-application-property [page 1541]; start-maintenance [page 1560]; stop-maintenance [page 1565]; rolling-update [page 1534]; hot-update [page 1473]; clear-downtime-app [page 1375]; set-downtime-app [page 1547]
Logging	list-logs [page 1496]; get-log [page 1448] list-loggers [page 1495]; set-log-level [page 1549]; reset-log-levels [page 1526]
Monitoring	list-availability-check [page 1477]; create-availability-check [page 1383]; delete-availability-check [page 1401] list-jmx-checks [page 1492]; create-jmx-check [page 1390]; delete-jmx-check [page 1410] list-alert-recipients [page 1479]; set-alert-recipients [page 1538]; clear-alert-recipients [page 1373]
Keystore	list-keystores [page 1493]; upload-keystore [page 1581]; download-key-store [page 1439]; delete-keystore [page 1413];
Connectivity	put-destination [page 1517] get-destination [page 1445] delete-destination [page 1406]

Group	Commands
SAP HANA / SAP ASE	list-application-datasources [page 1475]; list-dbms [page 1487]; list-dbs [page 1488]; list-schemas [page 1502]
	create-schema [page 1392]; bind-schema [page 1370]; unbind-schema [page 1572]; delete-schema [page 1415]; display-schema-info [page 1437]
	bind-hana-dbms [page 1368]; unbind-hana-dbms [page 1571]
	create-db-ase [page 1384]; create-db-user-ase [page 1387]; set-db-user-password-ase [page 1546]; display-db-info [page 1435]; set-db-properties-ase [page 1544]; bind-db [page 1364]; unbind-db [page 1569]; delete-db-ase [page 1402]; delete-db-user-ase [page 1405]
	create-db-hana [page 1386]; set-db-properties-hana [page 1545]; start-db-hana [page 1559]; stop-db-hana [page 1563]; delete-db-hana [page 1404]copy-db-hana-certificates [page 1378]
	grant-schema-access [page 1452]; revoke-schema-access [page 1533]; list-schema-access-grants [page 1503]
	open-db-tunnel [page 1514]; close-db-tunnel [page 1376]
	restart-hana [page 1529]
	grant-db-tunnel-access [page 1452]; revoke-db-tunnel-access [page 1532]; list-db-tunnel-access-grants [page 1490]
	grant-db-access [page 1450]; list-db-access-permissions [page 1485]; revoke-db-access [page 1531]
Document Service	add-ecm-tenant [page 1358]; create-ecm-repository [page 1388]; delete-ecm-repository [page 1407]; display-ecm-repository [page 1434]; edit-ecm-repository [page 1441]; list-ecm-repositories [page 1491]; reset-ecm-key [page 1525]
Subaccounts and Entitlements	create-account [page 1379]; delete-account [page 1400]; list-accounts [page 1478]; set-quota [page 1550]
Subscription Management	subscribe [page 1566]; unsubscribe [page 1577]; list-subscribed-accounts [page 1507]; list-subscribed-applications [page 1509]
Application Domains	add-custom-domain [page 1361]; add-platform-domain [page 1363]; list-application-domains [page 1482]; remove-custom-domain [page 1523]; remove-platform-domain [page 1524]
Custom SSL	add-ca [page 1360]; list-cas [page 1483]; remove-ca [page 1522]; create-ssl-host [page 1395]; delete-ssl-host [page 1412]; list-ssl-hosts [page 1506]; set-ssl-host [page 1552]
	display-csr [page 1433]; generate-csr [page 1446]
	bind-domain-certificate [page 1367]; change-domain-certificate [page 1371]; delete-domain-certificate [page 1408]; list-domain-certificates [page 1489]; unbind-domain-certificate [page 1570]; upload-domain-certificate [page 1578]

Group	Commands
Reverse-Proxy	list-proxy-host-mappings [page 1499]; map-proxy-host [page 1513]; unmap-proxy-host [page 1575]
Resources	delete-resource (Beta) [page 1411]
System	version [page 1583]; sdk-upgrade [page 1538]
Extensions	hcmcloud-create-connection [page 1454]; hcmcloud-delete-connection [page 1456]; hcmcloud-disable-application-access [page 1458]; hcmcloud-display-application-access-status [page 1459]; hcmcloud-enable-application-access [page 1461]; hcmcloud-enable-role-provider [page 1463]; hcmcloud-get-registered-home-page-tiles [page 1464]; hcmcloud-import-roles [page 1466]; hcmcloud-list-connections [page 1468]; hcmcloud-register-home-page-tiles [page 1469]; hcmcloud-unregister-home-page-tiles [page 1471]
MTA	delete-mta [page 1414];deploy-mta [page 1429]; display-mta [page 1436] ; list-mtas [page 1497]; list-mta-operations [page 1498]; subscribe-mta [page 1568]
Virtual Machines	create-vm [page 1397]; delete-vm [page 1418]; list-vms [page 1510]; reboot-vm [page 1519] close-ssh-tunnel [page 1376]; list-ssh-tunnels [page 1506]; open-ssh-tunnel [page 1516] create-security-rule [page 1394]; delete-security-rule [page 1416]; list-security-rules [page 1505] register-access-point [page 1521]; unregister-access-point [page 1576] delete-volume [page 1419] create-volume-snapshot [page 1399]; delete-volume-snapshot [page 1421]; display-volume-snapshot [page 1438]; list-volume-snapshots [page 1512]; list-volumes [page 1511]

6.3.3.4.1 add-ecm-tenant

Adds a new tenant to a repository.

```
neo add-ecm-tenant --account <subaccount_technical_name>  --host <host> --user
<e-mail_or_user> --name <repository_name> --key <repository_key> --tenant
<tenant_name> --virus-scan <true/false>
```

Parameters

Required	
`-a, --account`	Subaccount technical name Specify an existing subaccount of which you are already a member. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions and Hosts Available for the Neo Environment [page 17]
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-n, --name`	Name of the repository `Type`: string
`-t, --tenant`	Tenant alias `Type`: string
`-k, --key`	Key of the repository `Type`: string

Optional	
`-v, --virus-scan`	Can be used to activate the virus scanner and check all incoming documents for viruses. `Default`: `true` `Type`: boolean → Recommendation For repositories that are used by untrusted users and or for unknown content, we recommend that you enable the virus scanner by setting this parameter to `true`. Enabling the virus scanner could impair the upload performance. If a virus is detected, the upload process for the document fails with a virus scanner exception.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Example

```
neo add-ecm-tenant --account sap --host hana.ondemand.com --user
<myemail@example.com> --name DemoRepository --key ecm_012345689  --tenant sap2 --
virus-scan true
```

This example adds the *sap2* tenant to the *DemoRepository* repository.

6.3.3.4.2 add-ca

Uploads a trusted CA certificate and adds it to a certificate authority (CA) bundle. If you don't have a CA bundle yet, it will be created automatically.

```
neo add-ca --account <subaccount_technical_name> --host <host> --user <e-
mail_or_user> --bundle <bundle_name> --location ./certificate_name.crt
```

To configure a CA bundle, run `set-ssl-host` using the `--ca-bundle` parameter. For more information, see set-ssl-host [page 1552].

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions and Hosts Available for the Neo Environment [page 17].
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name. `Type`: string
`--bundle`	Name of a new or existing bundle in which CAs will be added. You can have several CA bundles, but you can assign only one bundle to one SSL host. One bundle can hold up to 150 certificates. `Type`: string When creating a new bundle, the bundle name must start with a letter and can only contain lowercase letters (a-z), uppercase letters (A-Z), numbers (0-9), underscores (_), and hyphens (-).

Required	
`-l, --location`	Path to a file that contains one or more certificates of trusted CAs in PEM format.

Example

```
neo add-ca --account mysubaccount --host hana.ondemand.com --user
mymail@example.com --bundle mybundle --location ./mycert.crt
```

Related Information

list-cas [page 1483]
remove-ca [page 1522]
set-ssl-host [page 1552]
Managing Client Certificate Authentication for Custom Domains [page 1660]

6.3.3.4.3 add-custom-domain

Use this command to add a custom domain to an application URL. This will route the traffic for the custom domain to your application on SAP BTP.

```
neo add-custom-domain --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --custom-domain <custom_domain>
--application-url <app_url> --ssl-host <ssl_hostname>
```

Parameters

To list all parameters available for this command, execute **`neo help add-custom-domain`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required

`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions and Hosts Available for the Neo Environment [page 17]
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-e, --custom-domain`	Custom domain for accessing the application `Type`: string (hostname such as mydomain.com or shop.mydomain.com corresponding to the CN/SAN of the certificate)
`-i, --application-url`	The access point of the application on SAP BTP default domains (hana.ondemand.com, etc.) `Type`: string (hostname such as myapp.hana.ondemand.com or shop-mytenant.hana.ondemand.com) Query strings are not supported in the `--application-url` parameter and are ignored. For example, if you specify "mysubaccountmyapp.hana.ondemand.com/sites?idp=example", the "?idp=example" part will be ignored. **i Note** For SAP Cloud Integration applications, the application URL is formed differently. For more information, see Configuring Custom Domains for SAP Cloud Integration.
`-l, --ssl-host`	SSL host as defined with the `--name` parameter when created, or 'default' if not specified.

Optional

`--disable-application-url`	Allows you to disable the access to the platform URL, for example `hana.ondemand.com`, for subscribed applications with a URL of type `https://<application_name><provider_subaccount>-<consumer_subaccount>.<domain>`. The *<domain>* is the respective region host. For example, `us1.hana.ondemand.com`. **i Note** It may take up to one hour for this change to take effect. `Type`: takes no value If you do not explicitly use this parameter, your subscribed application will continue to be accessible via the default URL `hana.ondemand.com`.

Example

```
neo add-custom-domain --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --custom-domain www.example.com
--application-url mysubaccountmyapp-subscription.hana.ondemand.com --ssl-host
mysslhostname
```

Related Information

Add the Custom Domain [page 1655]
list-custom-domain-mappings [page 1484]
remove-custom-domain [page 1523]
Configuring Custom Domains [page 1649]
Configuring Custom Domains for SAP Cloud Integration

6.3.3.4.4 add-platform-domain

Adds a platform domain (under hana.ondemand.com) on which the application will be accessed.

```
neo add-platform-domain --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user> --host <host> --platform-domain
<platform_domain>
```

Parameters

To list all parameters available for this command, execute **`neo help add-platform-domain`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required	
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions and Hosts Available for the Neo Environment [page 17].
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-m, --platform-domain`	Platform domain under hana.ondemand.com The chosen platform domain will be parent domain in the absolute application domain. `Acceptable values`: • svc.{region host} • cert.{region host} For more information about the available region hosts, see Regions and Hosts Available for the Neo Environment [page 17].

Example

```
neo add-platform-domain --account mysubaccount --application myapp --user myuser
--host us1.hana.ondemand.com --platform-domain svc.us1.hana.ondemand.com
```

Related Information

Using Platform Domains [page 1666]
remove-platform-domain [page 1524]

6.3.3.4.5 bind-db

This command binds an SAP HANA tenant database or SAP ASE user database to a Java application using a data source.

You can only bind an application to an SAP HANA tenant database or SAP ASE user database if the application is deployed.

- Database in the same subaccount:

```
neo bind-db -a <subaccount_technical_name> -b <application_name> -h <host> -u
<e-mail_or_user> -i <database_ID> --db-user <database_user> --db-password
<database_user_password>
```

- Database in another accout (same global account):

> **i Note**
>
> To bind your application to a database that is owned by another subaccount of your global account, you need permission to use it. See Sharing Databases in the Same Global Account.

```
neo bind-db -a <subaccount_technical_name> -b <application_name> -h <host> -u
<e-mail_or_user> -i <owner_subaccount>:<database_ID> --db-user
<database_user> --db-password <database_user_password>
```

- Database in another subaccount:

```
neo bind-db -a <subaccount_technical_name> -b <application_name> -h <host> -u
<e-mail_or_user> --access-token
vm6431dhjcr2e3dbt0fk6jpzm2w7oo3q48yumf1c6uu8b9pt9z --db-user <database_user>
--db-password <database_user_password>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions and Hosts Available for the Neo Environment [page 17]
`-i, --id`	Database ID `Type`: string
`--access-token`	Identifies a database access permission. The access token and database ID parameters are mutually exclusive.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Required	
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`--db-password`	Password of the database user used to access the database
`--db-user`	Name of the database user used to access the database

Optional	
`-s, --data-source`	Data source name Default: <DEFAULT> Type: string (uppercase and lowercase letters, numbers, and the following special characters: `/`, `_`, `-`, `@`. Do not use special characters as first or last charachters of the data source name.)

Example

- Database in the same subaccount:

```
neo bind-db -a mysubaccount -b myapp -h hana.ondemand.com -u
mymail@example.com -i mydb --db-user MYDBUSER --db-password SECRET
```

- Database in another subaccount (same global account):

```
neo bind-db -a mysubaccount -b myapp -h hana.ondemand.com -u
mymail@example.com -i anothersubaccount:anothersubaccountsdb --db-user
MYDBUSER --db-password SECRET
```

- Database in another subaccount:

```
neo bind-db -a mysubaccount -b myapp -h hana.ondemand.com -u
mymail@example.com --access-token
120579jy40i15v1dqv3n3fsw40ug52m6re9fzqxg46l3fah0w0 --db-user MYDBUSER --db-
password SECRET
```

Related Information

unbind-db [page 1569]
Binding SAP HANA Databases to Java Applications
Binding SAP ASE Databases to Java Applications
Binding SAP HANA Databases to Java Applications
Bind Applications to Databases in the Same Global Account
Bind Applications to Databases in Other Subaccounts

6.3.3.4.6 bind-domain-certificate

Binds a certificate to an SSL host. The certificate must already be uploaded.

```
neo bind-domain-certificate --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --ssl-host <ssl_hostname> --certificate
<certificate_name>
```

Parameters

To list all parameters available for this command, execute **`neo help bind-domain-certificate`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions and Hosts Available for the Neo Environment [page 17]
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or username. `Type`: string
`--certificate`	Name of the certificate that you set to the SSL host The certificate must already be uploaded.
`-l, --ssl-host`	SSL host as defined with the `--name` parameter when created, or 'default' if not specified.

Example

```
neo bind-domain-certificate --account mysubaccount --user mymail@example.com --
host hana.ondemand.com --ssl-host mysslhostname --certificate myfirstcert
```

Related Information

Bind the Certificate to the SSL Host [page 1654]
Configuring Custom Domains [page 1649]
set-ssl-host [page 1552]

6.3.3.4.7 bind-hana-dbms

This command binds a Java application to an SAP HANA single-container database system (XS) via a data source.

You can only bind an application to an SAP HANA single-container database system (XS) if the application is deployed.

The following commands are available:

- Database in the same subaccount:

```
neo bind-hana-dbms -a <subaccount_technical_name> -b <application_name> -h
<host> -u <e-mail_or_user> -i <productive_HANA_database> --db-user
<database_user> --db-password <database_user_password>
```

- Database in another subaccount:

```
neo bind-hana-dbms -a <subaccount_technical_name> -b <application_name> -h
<host> -u <e-mail_or_user> --access-token
120579jy40i15v1dqv3n3fsw40ug52m6re9fzqxg46l3fah0w0 --db-user <database_user>
--db-password <database_user_password>
```

> **i Note**
>
> To bind your application to a database that is owned by another subaccount of your global account, see bind-db [page 1364].

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required

`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions and Hosts Available for the Neo Environment [page 17] **i Note** You cannot use this command in trial accounts.
`-i, --id`	ID of the SAP HANA single-container database system (XS) `Type`: string
`--access-token`	Identifies a database access permission. The access token and database ID parameters are mutually exclusive.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`--db-password`	Password of the database user used to access the SAP HANA single-container database system (XS)
`--db-user`	Name of the database user used to access the SAP HANA single-container database system (XS)

Optional

`-s, --data-source`	Data source name Type: string (uppercase and lowercase letters, numbers, and the following special characters: `/`, `_`, `-`, `@`. Do not use special characters as first or last charachters of the data source name.)

Example

- Database in the same subaccount:

```
neo bind-hana-dbms -a mysubaccount -b myapp -h hana.ondemand.com -u
mymail@example.com -i myhanaxs --db-user MYPRODHANA --db-password SECRET
```

- Database in another subaccount:

```
neo bind-hana-dbms -a mysubaccount -b myapp -h hana.ondemand.com -u
mymail@example.com --db-user MYPRODHANA --db-password SECRET --access-token
120579jy40i15v1dqv3n3fsw40ug52m6re9fzqxg46l3fah0w0
```

Related Information

unbind-hana-dbms [page 1571]
Binding SAP HANA Databases to Java Applications
Bind Applications to Databases in the Same Global Account
Bind Applications to Databases in Other Subaccounts

6.3.3.4.8 bind-schema

This command binds a schema to a Java application via a data source. If a data source name is not specified, the schema will be automatically bound to the default data source of the application.

You can only bind a schema to an application if the application is deployed.

```
neo bind-schema -a <subaccount_technical_name> -b <application_name> -h <host> -
u <e-mail_or_user> -i <schema_ID>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions and Hosts Available for the Neo Environment [page 17]
`-i, --id`	Schema ID `Type`: string
`--access-token`	Identifies a schema access grant. The access token and schema ID parameters are mutually exclusive.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Required	
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Optional	
`-s, --data-source`	Data source name The application will be able to access the schema via the specified data source. Type: string (uppercase and lowercase letters, numbers, and the following special characters: `/`, `_`, `-`, `@`. Do not use special characters as first or last charachters of the data source name.)

Example

```
neo bind-schema -a mysubaccount -b myapp -h hanatrial.ondemand.com -u
mymail@example.com -i myschema -s datasource1
```

Related Information

Example Scenarios
Bind Schemas
grant-schema-access [page 1452]
unbind-schema [page 1572]
bind-hana-dbms [page 1368]
unbind-hana-dbms [page 1571]

6.3.3.4.9 change-domain-certificate

Changes a certificate of an SSL host. The certificate must already be uploaded.

```
neo change-domain-certificate --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --ssl-host <ssl_hostname> --certificate
<certificate_name>
```

Parameters

To list all parameters available for this command, execute **`neo help change-domain-certificate`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions and Hosts Available for the Neo Environment [page 17].
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`--certificate`	Name of the certificate that you set to the SSL host The certificate must already be uploaded.
`-l, --ssl-host`	SSL host as defined with the `--name` parameter when created, or 'default' if not specified.

Example

```
neo change-domain-certificate --account mysubaccount --user mymail@example.com --
host hana.ondemand.com --ssl-host mysslhostname --certificate myfirstcert
```

The `change-domain-certificate` command lets you change the domain certificate of a custom domain in one step instead of executing both the `unbind-domain-certificate` and `bind-domain-certificate` commands.

If your current version of the SDK for SAP BTP, Neo environment does not support this command, update your SDK or use the `unbind-domain-certificate` and `bind-domain-certificate` commands instead.

> **i Note**
>
> The first version of the SDK for SAP BTP, Neo environment to support the `change-domain-certificate` command are:
>
> - Java Web SDK (version 1.126.11)
> - Java EE 6 Web Profile SDK (version 2.108.10)
> - Java Web Tomcat 7 SDK (version 2.73.15)

- Java Web Tomcat 8 SDK (version 3.27.14)

For more information, see Update the SAP BTP SDK for Neo Environment [page 849].

Related Information

Update an Expired Certificate [page 1660]
unbind-domain-certificate [page 1570]
bind-domain-certificate [page 1367]
set-ssl-host [page 1552]
Update the SAP BTP SDK for Neo Environment [page 849]

6.3.3.4.10 clear-alert-recipients

Clears alert recipients.

Prerequisites

- You have set up the console client.
  For more information, see Set Up the Console Client [page 847].
- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Overview

- Clearing alert recipients becomes effective after five minutes.
- If no emails are specified, it clears all recipients.
- Clearing alert recipients and setting them again triggers emails for confirmation to the recipients one more time.

```
neo clear-alert-recipients -a <subaccount_technical_name> -u <e-mail_or_user> -h
<host>
```

Parameter

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID, or user name `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions and Hosts Available for the Neo Environment [page 17]

Optional	
`-b, --application`	Application name for Java or HTML5 applications, or productive SAP HANA instance database name and application name in the format `<instance name>:<application name>` for SAP HANA XS applications `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-e, --email`	Comma-separated list of recipient e-mails `Type`: string

Example

```
neo clear-alert-recipients -a mysubaccount -b demo -u p1234567 --host
hana.ondemand.com
```

Related Information

Monitoring Java Applications [page 729]
Monitoring Database Systems [page 760]
list-alert-recipients [page 1479]
set-alert-recipients [page 1538]

6.3.3.4.11 clear-downtime-app

The command deregisters a previously configured downtime page for an application. After you execute the command, the default HTTP error will be shown to the user in the event of unplanned downtime.

```
neo clear-downtime-app --account <subaccount_technical_name> --application
<application_name> --host <host>
--user <e-mail_or_user>
```

Parameters

To list all parameters available for this command, execute **`neo help clear-downtime-app`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions and Hosts Available for the Neo Environment [page 17]
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

```
neo clear-downtime-app --account mysubaccount --application myapp --user
<mymail@example.com
```

Related Information

set-downtime-app [page 1547]

6.3.3.4.12 close-db-tunnel

This command closes one or all database tunnel sessions that have been opened in a background process using the `open-db-tunnel --background` command.

```
neo close-db-tunnel --session-id <session_ID>
```

A tunnel opened in a background process is automatically closed when the last session using the tunnel is closed. The background process terminates after the last tunnel has been closed.

Parameters

Required	
`--all`	Closes all tunnel sessions that have been opened in the background
`--session-id`	Tunnel session to be closed. Cannot be used together with the parameter `--all`.

Example

```
neo close-db-tunnel --session-id f4b00f06-df0a-4018-b725-392a93b49bd4
```

Related Information

open-db-tunnel [page 1514]
Automating the Use of Database Tunnels

6.3.3.4.13 close-ssh-tunnel

Closes the ssh-tunnel to the specified virtual machine. If no virtual machine ID is specified, closes all tunnels.

```
neo close-ssh-tunnel --vm-id <vm_id>
```

or

```
neo close-ssh-tunnel --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --vm-name <vm_name>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) `Condition`: Required if you use `--vm-name`
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions and Hosts Available for the Neo Environment [page 17]. `Condition`: Required if you use `--vm-name`
`-n, --vm-name`	Name of the virtual machine `Type`: string `Condition`: Use either `--vm-id` or `--vm-name` `Condition`: `--vm-name` requires the following parameters: • `-h, --host` • `-u, --user` • `-a, --account` • `-p, --password`
`-i ,--vm-id`	ID of the virtual machine `Type`: string If no virtual machine ID is specified, closes all tunnels. `Condition`: Use either `--vm-id` or `--vm-name`
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string `Condition`: Required if you use `--vm-name`
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string `Condition`: Required if you use `--vm-name`

Optional	
`-r, --port`	Port on which you want to close the SSH tunnel

Example

```
neo close-ssh-tunnel --vm-id myvmid --port xxxx
```

or

```
neo close-ssh-tunnel --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --vm-name myvmname --port xxxx
```

Related Information

open-ssh-tunnel [page 1516]
list-ssh-tunnels [page 1506]

6.3.3.4.14 copy-db-hana-certificates

This command copies certificates from the file-based certificate store 'sapsrv.pse' to the SAML and X509 certificate collections of the in-database certificate store of an SAP HANA tenant database.

```
neo copy-db-hana-certificates -a <subaccount_technical_name> -h <host> -u <e-
mail_or_user> -i <database_ID> --db-user <dbuser> --db-password
<database_user_password>
```

Parameters

Required	
`-i, --id`	HANA database ID `Type`: string **i Note** The tenant database must be started.

Required	
, `--db-user`	Name of the SAP HANA tenant database user used for the operation `Type`: string **i Note** The database user must have the following privileges in the HANA tenant database: "USER ADMIN", "CERTIFICATE ADMIN", "TRUST ADMIN".
`--db-password`	Password of the SAP HANA tenant database user (optional, queried at the command prompt if omitted)

Example

```
neo copy-db-hana-certificates -a mysubaccount -h hana.ondemand.com -u
mymail@example.com -i mydb --db-user mydbuser --db-password SECRET
```

Related Information

Copying Certificates to In-Database Certificate Store

6.3.3.4.15 create-account

Creates a new subaccount with an automatically generated unique ID as subaccount technical name and the specified display name and assigns the user as a subaccount owner. The user is authorized against the existing subaccount passed as --account parameter. Optionally, you can clone an existing subaccount configuration to save time and effort.

```
neo create-account --display-name <subaccount_display_name> --account
<subaccount_technical_name> --user <e-mail_or_user> --host <host> --clone
<cloning_options>
```

i Note

If you clone an existing extension subaccount, the new subaccount will **not** be an extension subaccount but a regular one. The new subaccount will **not** have the trust and destination settings typical for extension subaccounts.

Parameters

To list all parameters available for this command, execute **`neo help create-account`** in the command line.

Required	
`-a`, `--account`	Subaccount technical name Specify an existing subaccount of which you are already a member. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-n`, `--display-name`	Subaccount display name If you want to create a subaccount whose display name has intervals, use quotes when executing the command. For example: neo ... --display-name "Display Name with Intervals" `Type`: string (up to 255 characters)
`-u`, `--user`	Use your email, SAP ID or user name `Type`: string
`-p`, `--password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-h`, `--host`	Enter a region host. `Type`: URL. For acceptable values see Regions and Hosts Available for the Neo Environment [page 17]
--clone	(Optional) List of settings that will be copied (re-created) from the existing subaccount into the new subaccount. A comma separated list of values, which are as follows: • `trust` • `members` • `destinations` • `all` Each value combination is acceptable, for example, `trust,members` or `all`. If specified, all the configurations of the passed type(s) will be cloned to the newly created subaccount. **→ Tip** We recommend listing explicitly the required cloning options instead of using `--clone all` in automated scripts. This will ensure backward compatibility in case the available cloning options, enveloped by `all`, change in future releases.

Example

```
neo create-account --account mysubaccount --display-name mynewsubaccount --user
myuser --host hana.ondemand.com
```

Cloning Existing Subaccounts

You may choose among the following cloning options:

Cloning Option	Description
`all`	All settings (trust, members and destinations) from the existing subaccount will be copied into the new one. **⚠ Caution** The list of cloned configurations might be extended in the future.

Cloning Option	Description
`trust`	The following trust settings will be re-created in the new subaccount similarly to the relevant settings in the existing subaccount: • Trusted Identity Authentication tenants - the new subaccount will have trust to the Identity Authentication tenants registered in the existing subaccount. The respective Identity Authentication tenants, in turn, will automatically register a new service provider corresponding to the new SAP BTP subaccount. > **Note** > SAP BTP will generate a new pair of key and certificate on behalf of the new subaccount. Remember to replace them with your proprietary key and certificate when using the subaccount for productive purposes. All other trust settings (for example, trusted on-premise identity providers) from the existing subaccount will **not** be copied into the new subaccount. > **Note** > If you do not have any trusted Identity Authentication tenants in the existing subaccount, cloning the trust settings will result in trust with SAP ID Service (as default identity provider) in the new subaccount.
`members`	All members with their roles from the existing subaccount will be copied into the new one.
`destinations`	All destinations from the existing subaccount will be created into the new one. In addition, the relevant certificates and passwords for the destinations will also be cloned so the destination configurations will be fully functional in the new subaccount.

Example of cloning an existing subaccount to create a new subaccount with the same trust settings and existing destinations:

```
neo create-account --account mysubaccount --display-name mynewsubaccount --user
myuser --host hana.ondemand.com --clone trust,destinations
```

6.3.3.4.16 create-availability-check

Creates an availability check.

```
neo create-availability-check -a <subaccount_technical_name> -u <e-mail_or_user>
-U <relative_URL> -h <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-U, --url`	Relative application URL `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions

Optional	
`-b, --application`	Application name for Java or HTML5 applications, or productive SAP HANA instance database name and application name in the format `<instance name>:<application name>` for SAP HANA XS applications `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-W, --warning`	Threshold value or range of the response time in seconds. `Default`: 50 `Type`: string
`-C , --critical`	Threshold value or range of the response time in seconds. `Default`: 60 `Type`: string
`-w, --overwrite`	Should be used only if there is an existing alert that needs to be updated. `Default`: false `Type`: boolean

Example

Example for creating an availability check for application *demo*:

```
neo create-availability-check -a mysubaccount -b demo -u p1234567 -U /heartbeat -
C 4 -W 6 --host hana.ondemand.com
```

Example for creating an availability check for *myhana* application:

```
neo create-availability-check -a mysubaccount -b myhanainstance:myhana -u
p1234567 -U /heartbeat.xsjs -C 4 -W 6 --host hana.ondemand.com
```

Related Information

Monitoring Java Applications [page 729]
Monitoring Database Systems [page 760]

6.3.3.4.17 create-db-ase

This command creates an ASE database with the specified ID and settings on an ASE database system.

```
neo create-db-ase -a <subaccount_technical_name> -h <host> -u <e-mail_or_user> --
dbsystem <database_system> -i <database_ID> --db-user <dbuser> --db-password
<database_user_password> --db-size <database_size>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Required	
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-i, --id`	ASE database ID `Type`: string
`--dbsystem`	ID of a productive ASE database system `Type`: string
`--db-user`	Name of the user for the ASE database `Type`: string (up to 30 characters, starting with a letter)
`--db-password`	Password of the database user used to access the ASE database (optional, queried at the command prompt if omitted)
`--db-size`	Size of the database in MB **i Note** This parameter sets the maximum database size. The minimum database size is 24 MB. You receive an error if you enter a database size that exceeds the quota for this database system. The size of the transaction log will be at least 25% of the database size you specify.

i Note

There is a limit to the number of databases you can create, and you'll see an error message when you reach the maximum number of databases. For more information on user database limits, see Creating Databases.

Example

```
neo create-db-ase -a mysubaccount -h hana.ondemand.com -u mymail@example.com --
dbsystem mydbsys -i mydb --db-user mydbuser --db-password SECRET --db-size
mydbsize
```

Related Information

Create SAP ASE Tenant Databases

delete-db-ase [page 1402]

6.3.3.4.18 create-db-hana

This command creates an SAP HANA database with the specified ID and settings, on an SAP HANA tenant database system.

```
neo create-db-hana -a <subaccount_technical_name> -h <host> -u <e-mail_or_user>
--dbsystem <database_system> -i <database_ID> --db-password
<database_user_password>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-i, --id`	HANA database ID `Type`: string
`--dbsystem`	ID of the SAP HANA database system `Type`: string
`--db-password`	Password of the SYSTEM user used to access the SAP HANA database (optional, queried at the command prompt if omitted)

Optional	
`--dp-server`	Enables or disables the data processing server of the SAP HANA database: 'enabled', 'disabled' (default).
`--script-server`	Enables or disables the script server of the SAP HANA database: 'enabled', 'disabled' (default).
`--web-access`	Enables or disables access to the SAP HANA database from the Internet: 'enabled' (default), 'disabled'
`--xsengine-mode`	Specifies how the XS engine should run: 'embedded' (default), 'standalone'.

> **i Note**
>
> There is a limit to the number of databases you can create, and you'll see an error message when you reach the maximum number of databases. For more information on tenant database limits, see Creating Databases.

Example

```
neo create-db-hana -a mysubaccount -h hana.ondemand.com -u mymail@example.com --
dbsystem mydbsys -i mydb --db-password SECRET
```

Related Information

Create SAP HANA Tenant Databases
delete-db-hana [page 1404]

6.3.3.4.19 create-db-user-ase

This command creates a user of an ASE database.

```
neo create-db-user-ase -a <subaccount_technical_name> -h <host> -u <e-
mail_or_user> -i <database_ID> --db-user <dbuser> --db-password
<database_user_password>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Required	
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-i, --id`	ASE database ID `Type`: string
`--db-user`	Name of the ASE database user `Type`: string (up to 30 characters, starting with a letter)
`--db-password`	Password of the ASE database user (optional, queried at the command prompt if omitted)
`--schema-user`	(optional) The user will have a schema scope and will be created with its own schema on the tenant database; if omitted, the user will have a database scope and no own schema.

Example

```
neo create-db-user-ase -a mysubaccount -h hana.ondemand.com -u
mymail@example.com -i mydb --db-user mydbuser --db-password SECRET
```

6.3.3.4.20 create-ecm-repository

Creates a new repository in the specified subaccount.

```
neo create-ecm-repository --account <subaccount_technical_name>  --host <host> --
user <e-mail_or_user> --name <repository_name> --key <repository_key> --display
<display_name_of_repository> --description <description_of_repository> --virus-
scan <true/false>
```

Parameters

Required	
`-a, --account`	Subaccount technical name Specify an existing subaccount of which you are already a member. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required

`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-n, --name`	Name of the repository `Type`: string
`-k, --key`	Key of the repository `Type`: string

Optional

`-d, --display-name`	Can be used to provide a more readable name of the repository. Equals the --name value if left blank. You cannot change the display later on. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-e, --description`	Description of the repository. You cannot change the description later on. `Type`: string
`-v, --virus-scan`	Can be used to activate the virus scanner and check all incoming documents for viruses. `Default`: `true` `Type`: boolean → Recommendation For repositories that are used by untrusted users and or for unknown content, we recommend that you enable the virus scanner by setting this parameter to `true`. Enabling the virus scanner could impair the upload performance. If a virus is detected, the upload process for the document fails with a virus scanner exception.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Example

```
neo create-ecm-repository --account sap --host hana.ondemand.com --user
<myemail@example.com> --name DemoRepository --key ecm_012345689  --display-name
DemoRep --description "Demo Repository" --virus-scan true
SAP BTP Console Client
```

```
Repository DemoRepository created successfully.
```

6.3.3.4.21 create-jmx-check

Creates a JMX check.

```
neo create-jmx-check -a <subaccount_technical_name> -u <e-mail_or_user> -n
<JMX_check_name> -O <MBean_object_name> -A <MBean_object_attribute> -h <host>
```

Parameters

> **i Note**
>
> The JMX check settings support the JMX specification. For more information, see Java Management Extensions (JMX) Specification .

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-n, --name`	Name of the JMX check The name must be up to 99 characters long and must not contain the following symbols: ` ~ ! $ % ^ & * | ' " < > ? , ( ) = `Type`: string
`-O, --object-name`	Object name of the MBean that you want to call If it contains quotation marks, they should be escaped with '\'. `Type`: string
`-A, --attribute`	Name of the attribute inside the class with the specified object name. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions

Optional	
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) **i Note** If the parameter is not used, the JMX check will be on subaccount level for all running applications in the subaccount.
`-K, --key`	Attribute key It is needed only if the attribute is a composite data structure. This key defines the item in the composite data structure. For more information about the composite data structure, see Class CompositeDataSupport . `Type`: string
`-o, --operation`	Operation that has to be called on the MBean after checking the attribute value. It is useful for resetting statistical counters to restart an operation on the same MBean. `Type`: string
`-U, --unit`	Unit of measurement `Type`: string
`-W, --warning`	Warning threshold The threshold can be a regular expression in case of string values or compliant with the official nagios threshold/ranges format. For more information about the format in case it is a number, see the official nagios documentation .
`-C , --critical`	Critical threshold The threshold can be a regular expression in case of string values or compliant with the official nagios threshold/ranges format. For more information about the format in case it is a number, see the official nagios documentation .
`-w, --overwrite`	Overwrites an existing check. `Default`: false `Type`: boolean **i Note** When you use this parameter, a new JMX check is created if the one you specify does not exist.

Example 1: Configuring a JMX Check

For a typical example how to configure a JMX check for your application and subscribe recipients to receive notification alerts, see Configure JMX Checks for Java Applications from the Console Client [page 741].

Example 2: Using Warning and Critical Thresholds

The following example creates a JMX check that returns a warning state of the metric if the value is between 10 and 100 bytes, and returns a critical state if the value is greater than 100 bytes. If the value is less than 10 bytes, the returned state is OK.

```
neo create-jmx-check -a mysubaccount -b demo -u p1234567 -n "JVM Heap Memory
Used" -O java.lang:type=Memory -A HeapMemoryUsage -K used -U B -W 10 -C 100 -h
hana.ondemand.com
```

Example 3: Using an Operation on an MBean

```
neo create-jmx-check -a mysubaccount -b demo -u p1234567 -n "JVM Heap Memory
Used" -O Catalina:type=GlobalRequestProcessor,name="http-bio-8041" -A
HeapMemoryUsage -o resetCounters -h hana.ondemand.com
```

Related Information

JMX Checks [page 738]
Monitoring Java Applications [page 729]

6.3.3.4.22 create-schema

This command creates a HANA database or schema with the specified ID on a shared or dedicated database.

> ⚠ Caution
>
> This command is not supported for productive SAP HANA database systems. For more information about how to create schemas on productive SAP HANA database systems, see Binding SAP HANA Databases to Java Applications.

```
neo create-schema --account <subaccount_technical_name> --host <host> --id
<schema_ID> --user <e-mail_or_user> --dbtype <database_type>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-d, --dbtype`	Creates the HANA database or schema on a shared database system. Syntax: 'type:version'. Version is optional. Available database types: 'MaxDB', 'HANA', and 'HANAXS' (case-insensitive) To see which versions are available, execute the `list-dbms` command. `Type`: string
`--dbsystem`	Creates the schema on a dedicated database system. To see the available dedicated database systems, execute the `list-dbms` command. `Type`: string **⚠ Caution** The `list-dbms` command lists different database types, including productive SAP HANA database systems. Do not use the `create-schema` command for productive SAP HANA database systems. For more information about how to create schemas on productive SAP HANA database systems, see Binding SAP HANA Databases to Java Applications.
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-i, --id`	HANA database or schema ID It must start with a letter and can contain lowercase letters ('a' - 'z') and numbers ('0' - '9'). For schemas IDs, uppercase letters ('A' - 'Z') and the special characters '.' and '-' are also allowed. The ID must be unique within the subaccount. Note that the actual ID assigned in the database will be different to this version. `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Example

```
neo create-schema --account mysubaccount --host hanatrial.ondemand.com -i
myschema --user mymail@example.com --dbtype hana
```

Related Information

Example Scenarios
Administering Database Schemas

6.3.3.4.23 create-security-rule

This console client command creates a security group rule for a virtual machine.

```
neo create-security-rule --account <subaccount_technical_name> --host <host> --
user <e-mail_or_user> --name <vm_name> --source-id <system_ID> --source-type
<source_type> --from-port <from_port> --to-port <to_port>
```

Parameters

Required	
`-a, --account`	Your subaccount technical name. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-n, --name`	Type the name of the virtual machine. `Type`: string `Condition`: Use either `--id` or `--name`
`-i , --id`	ID of the virtual machine `Type`: string `Condition`: Use either `--id` or `--name`

Required	
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name. `Type`: string
`--from-port`	The start of the range of allowed ports. The <from_port> value must be less than or equal to the <to_port> value. `Type`: positive integer between 1 and 65535
`--to-port`	The end of the range of allowed ports. The <to_port> value must be greater than or equal to the <from_port> value. `Type`: positive integer between 1 and 65535
`--source-id`	The name of the system that you want to connect from. For an SAP HANA system, the `--source-id` is the SAP HANA database system name. For a Java application, it is the application name.
`--source-type`	The type of the system specified in the source ID. `Acceptable values`: HANA, JAVA

Example

```
neo create-security-rule --account mysubaccount --host hana.ondemand.com --user
mymail@example.com --name myvm --source-id mysystem --source-type HANA --from-
port <from_port> --to-port <to_port>
```

Related Information

Manage Network Communication for SAP Virtual Machine Service

6.3.3.4.24 create-ssl-host

Creates an SSL host for configuration of custom domains. This SSL host will be serving your custom domain.

```
neo create-ssl-host --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --name <ssl_host_name>
```

Parameters

To list all parameters available for this command, execute **`neo help create-ssl-host`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or username. `Type`: string

Optional	
`-n, --name`	Unique identifier of the SSL host. If not specified, 'default' value is set. `Type`: string When creating a new SSL host, the SSL host name must start with a letter and can only contain lowercase letters (a-z), uppercase letters (A-Z), numbers (0-9), underscores (_), and hyphens (-).

Example

```
neo create-ssl-host --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --name mysslhostname
```

> **i** Note
>
> In the command output, you get the SSL host. For example, "`A new SSL host [mysslhostname] was created and is now accessible on host [123456.ssl.ondemand.com]`". Write down the `123456.ssl.ondemand.com` host as you will later need it for the DNS configuration.

Related Information

Create an SSL Host [page 1651]

Configuring Custom Domains [page 1649]

6.3.3.4.25 create-vm

Creates a new virtual machine and starts it.

```
neo create-vm --account <subaccount_technical_name> --user <e-mail_or_user> --
host <host> --name <vm_name> --size <vm_size>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-n, --name`	Name of the virtual machine `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-z, --size`	The size of the newly created virtual machine. `Acceptable values`: x-small, small, medium, large, x-large

Optional	
`--preserve-volume`	Choose to preserve the volume on virtual machine termination. `Type`: switch. It takes no value. `Default`: off

Optional

`--private-key-passphrase`	Private key passphrase The passphrase is used to encrypt the private key in the generated key pair. You will need it when you connect to the virtual machine using an SSH client. > **i Note** > The passphrase should contain at least five characters in total: lowercase letters (a-z), uppercase letters (A-Z), numbers (0-9), and special characters. For production scenarios, make sure to protect your key pair with a passphrase that consists of at least fifteen characters. If you do not provide `-pkp` as a parameter in the command line, you will be prompted to enter a passphrase. If you do not enter a passphrase, the command will be executed but the private key will not be encrypted.
`--private-key-passphrase-confirmation`	Private key passphrase confirmation If you do not provide `-pkpc` as a parameter in the command line, you will be prompted to confirm your passphrase.
`-l, --ssh-key-location`	The path to a public key of certificate that will be uploaded and used to log in on the newly created virtual machine. `Type:` string
`-k, --ssh-key-name`	The name of the already existing public key to be used to login on the newly created virtual machine. `Type:` string. It can contain only alphanumeric characters (0-9, a-z, A-Z), underscore (_), and hyphen (-). The allowed name length is between 1 and 128 characters.
`-v, --volume-id`	Unique identifier of the volume from which the virtual machine will be created. `Type:` string `Condition`: Use when you want to create a new virtual machine from a volume.
`-s, --volume-snapshot-id`	Unique identifier of the volume snapshot. `Type:` string `Condition`: Use when you want to create a new virtual machine from a volume snapshot.
`-y, --synchronous`	Waits until the creation is complete. `Type:` switch. It takes no value. `Default`: off

Example

```
neo create-vm --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --name myvm --size x-small
```

Related Information

Manage Virtual Machines
Manage Volume Snapshots

6.3.3.4.26 create-volume-snapshot

Takes a snapshot of the file system of the specified virtual machine volume. The operation is asynchronous.

```
neo create-volume-snapshot --volume-id <volume_id> --name <name_of_the_snapshot>
--account <subaccount_technical_name> --user <e-mail_or_user> --host <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-n, --name`	Name of the snapshot `Type`: string. It can contain only alphanumeric characters (0-9, a-z, A-Z), underscore (_), and hyphen (-). The allowed name length is between 1 and 128 characters.
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Required	
`-v, --volume-id`	Unique identifier of the volume from which the snapshot will be taken `Type`: string

Example

```
neo create-volume-snapshot --volume-id myvolumeid --name myvolumesnapshot --
account mysubaccount --host hana.ondemand.com --user myemail@example.com
```

Related Information

Manage Volume Snapshots
delete-volume-snapshot [page 1421]
display-volume-snapshot [page 1438]
list-volume-snapshots [page 1512]

6.3.3.4.27 delete-account

Deletes a particular subaccount. Only the user who has created the subaccount is allowed to delete it.

> **i** Note
>
> You cannot delete a subaccount if it still has associated services, subscriptions, non-shared database systems, database schemas, deployed applications, HTML5 applications, or document service repositories. You need to disable services and delete the other subaccount entities before you proceed with the subaccount deletion. For information on how to disable services and delete subaccount entities, see Related Information. Make sure also that there are no running virtual machines in the subaccount.

```
neo delete-account --account <subaccount_technical_name> --user <e-mail_or_user>
--host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help delete-account`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.

Example

```
neo delete-account --account mysubaccount --user myuser --host hana.ondemand.com
```

Related Information

Disable Services in the Neo Environment [page 1154]
Deleting Subscriptions [page 1577]
Deleting Databases
Deleting Database Schemas [page 1415]
Undeploying Applications [page 1574]

6.3.3.4.28 delete-availability-check

Deletes an availability check.

```
neo delete-availability-check -a <subaccount_technical_name> -u <e-mail_or_user>
-h <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions

Optional	
`-b, --application`	Application name for Java or HTML5 applications, or productive SAP HANA instance database name and application name in the format `<instance name>:<application name>` for SAP HANA XS applications `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Example

```
neo delete-availability-check -a mysubaccount -b demo -u p1234567 --host
hana.ondemand.com
```

Related Information

Monitoring Java Applications [page 729]
Monitoring Database Systems [page 760]

6.3.3.4.29 delete-db-ase

This command deletes the ASE database with the specified ID.

```
neo delete-db-ase -a <subaccount_technical_name> -h <host> -u <e-mail_or_user> -
i <database_ID>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-i, --id`	ASE database ID `Type`: string

Optional	
`--force or -f`	Forcefully deletes the ASE database, including all application bindings
`--silent`	Suppresses the command line confirmation prompt

Example

```
neo delete-db-ase -a mysubaccount -h hana.ondemand.com -u mymail@example.com -i
mydb
```

Related Information

create-db-ase [page 1384]

6.3.3.4.30 delete-db-hana

This command deletes the SAP HANA database with the specified ID on a SAP HANA database system enabled for multitenant database container support.

```
neo delete-db-hana -a <subaccount_technical_name> -h <host> -u <e-mail_or_user> -
i <database_ID>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-i, --id`	HANA database ID `Type`: string

Optional	
`--force or -f`	Forcefully deletes the HANA database, including all application bindings
`--silent`	Suppresses the command line confirmation prompt

Example

```
neo delete-db-hana -a mysubaccount -h hana.ondemand.com -u mymail@example.com -i
mydb
```

6.3.3.4.31 delete-db-user-ase

This command deletes a user from an ASE database.

```
neo delete-db-user-ase -a <subaccount_technical_name> -h <host> -u <e-
mail_or_user> -i <database_ID> --db-user <dbuser>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-i, --id`	ASE database ID `Type`: string
`--db-user`	Name of the user for the ASE database

Optional	
`--silent`	Suppresses the command line confirmation prompt

Example

```
neo delete-db-user-ase -a mysubaccount -h hana.ondemand.com -u
mymail@example.com -i mydb --db-user mydbuser
```

Related Information

create-db-user-ase [page 1387]

6.3.3.4.32 delete-destination

This command deletes destination configuration properties files and JDK files. You can delete them on subaccount, application or subscribed application level.

```
neo delete-destination --account <subaccount_technical_name> --user <e-
mail_or_user> --name <destination_file_or_JKS_file> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help delete-destination`** in the command line.

Required	
`-a, --account`	Your subaccount. The subaccount for which you provide username and password. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	The application for which you delete a destination. Cases: • Use `--application <myapp>` if the application is running in your subaccount. • Use `--application <provider_subaccount>:<provider_app>` if the application is running in another subaccount. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`--name`	The name of the destination or JKS file to be deleted. `Type`: string
`-p, --password`	Password for the specified user. To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Your email, SAP ID or user name `Type`: string

Examples

- To delete a destination on subaccount level, execute:

```
neo delete-destination --account mysubaccount --user p1234567890 --name
myconfiguration.jks --host hanatrial.ondemand.com
```

- To delete a destination on application level, execute:

```
neo delete-destination --account mysubaccount --user p1234567890 --
application demo --name weather --host hanatrial.ondemand.com
```

- To delete a destination on subscribed application level, execute:

```
neo delete-destination --account mysubaccount --user p1234567890 --
application othersubaccount:remotedemo --name weather --host
hanatrial.ondemand.com
```

Related Information

Delete Destinations [page 61]
Exit Codes [page 1584]

6.3.3.4.33 delete-ecm-repository

This command deletes a repository including the data of any tenants in the repository, unless you restrict the command to a specific tenant.

> ⚠ Caution
>
> Be very careful when using this command. Deleting a repository permanently deletes all data. This data cannot be recovered.

Parameters

Required	
`-a, --account`	Subaccount technical name Specify an existing subaccount of which you are already a member. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required	
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-n, --name`	Name of the repository `Type`: string
`-k, --key`	Key of the repository `Type`: string

Optional	
`-t, --tenant`	Tenant alias Deletes the repository for the given tenant only instead of for all tenants. If no tenant name is provided, the repositories for all tenants are deleted. `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Example

```
neo delete-ecm-repository --account sap --host hana.ondemand.com --user
<myemail@example.com> --name DemoRepository --key ecm_012345689
SAP BTP Console Client
Are you sure you want to permanently delete all data? This operation cannot be
reverted. (yes/no)
yes
Delete command executed successfully.
```

6.3.3.4.34 delete-domain-certificate

Deletes a certificate.

> **Note**
>
> Cannot be undone. If the certificate is mapped to an SSL host, the certificate will be removed from the SSL host too.

```
neo delete-domain-certificate --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --name <certificate_name>
```

Parameters

To list all parameters available for this command, execute **`neo help delete-domain-certificate`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-n, --name`	Name of the certificate that you set to the SSL host The certificate must already be uploaded.

Example

```
neo delete-domain-certificate --account mysubaccount --user mymail@example.com --
host hana.ondemand.com --name myfirstcert
```

Related Information

Bind the Certificate to the SSL Host [page 1654]
bind-domain-certificate [page 1367]

6.3.3.4.35 delete-jmx-check

Deletes the specified JMX check or all JMX checks.

```
neo delete-jmx-check -a <subaccount_technical_name> -u <e-mail_or_user> -n
<JMX_check_name> -h <host>
```

or

```
neo delete-jmx-check -a <subaccount_technical_name> -u <e-mail_or_user> -A -h
<host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-n, --name` or `-A, all`	Name of the JMX check to be deleted or all JMX checks configured for the given subaccount and application are deleted. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions

Optional	
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Example

```
neo delete-jmx-check -a mysubaccount -b demo -u p1234567 -n "JVM Heap Memory
Used" -h hana.ondemand.com
```

Related Information

JMX Checks [page 738]
Monitoring Java Applications [page 729]

6.3.3.4.36 delete-resource (Beta)

Deletes a solution resource file from the system repository of a specified extension subaccount.

> **i Note**
>
> This is a beta feature available for SAP BTP extension subaccounts. For more information, see Important Disclaimers and Legal Information.

```
neo delete-resource --name <resource_name> --account <subaccount_technical_name>
--user <e-mail_or_user> --host <host> --silent
```

Parameters

To list all parameters available for this command, execute **`neo help delete-resource`** in the command line.

Required	
`-n, --name`	Name of the resource to be deleted `Type`: string
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-s, --silent`	Suppresses the command-line confirmation prompt

Example

To delete a solution resource from the system repository for your extension subaccount, execute:

```
neo delete-resource --name myresourcename --account myextensionsubacc --user
mymail@example.com --host hana.ondemand.com
```

6.3.3.4.37 delete-ssl-host

Deletes an SSL host.

```
neo delete-ssl-host --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --name <ssl_host_name>
```

Parameters

To list all parameters available for this command, execute **`neo help delete-ssl-host`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or username. `Type`: string
`-n, --name`	SSL host as defined with --create-ssl-host `Type`: string

Example

```
neo delete-ssl-host --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --name mysslhostname
```

Related Information

create-ssl-host [page 1395]
list-ssl-hosts [page 1506]

6.3.3.4.38 delete-keystore

This command is used to delete a keystore by deleting the keystore file. You can delete keystores on subaccount, application, and subscription levels.

Parameters

To list all parameters available for this command, execute **`neo help delete-keystore`** in the command line.

Required	
`-a, --account`	Consumer subaccount technical name The subaccount for which you provide username and password. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-n,--name`	Name of the keystore to be deleted `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-b, --application`	Application name • Use `--application <consumer_application_name>` if the application is running in your subaccount. • Use `--application <provider_subaccount_technical_name>:<provider_application_name>` if the application is running in another subaccount. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Example

On Subscription Level

```
neo delete-keystore --account <consumer_subaccount_technical_name> --application
<provider_subaccount_technical_name>:<provider_application_name>
--user <e-mail_or_user> --name KeyStore1 --host hana.ondemand.com
```

On Application Level

```
neo delete-keystore --account <consumer_subaccount_technical_name> --application
<consumer_application_name>
--user <e-mail_or_user> --name KeyStore1 --host hana.ondemand.com
```

On Subaccount Level

```
neo delete-keystore --account <consumer_subaccount_technical_name> --user <e-
mail_or_user> --name KeyStore1 --host hana.ondemand.com
```

Related Information

Keystore Console Commands [page 1783]
Keys and Certificates [page 1781]
Using the Keystore Service for Client Side HTTPS Connections [page 1791]

6.3.3.4.39 delete-mta

This command deletes Multitarget Application (MTA) archives that are deployed to your subaccount.

```
neo delete-mta --host <host> --account <subaccount_technical_name> --user <e-
mail_or_user> --id <MTA_ID>
```

Parameters

To list all parameters available for this command, execute **`neo help delete-mta`** in the command line.

Required	
`-a, --account`	The name of the subaccount for which you provide a user and a password.
`-h, --host`	The host on which you execute the command.
`-p, --password`	Your user password. We recommend that you enter it only when prompted, and not explicitly as a parameter in a properties file or the command line.

Required	
`-u, --user`	Your user e-mail or SAP ID (SCN) user name.
`-i, --id`	A comma-separated list of MTA IDs

Optional	
`-y, --synchronous`	Instructs the console to wait for the operation to finish. It takes no value.

Example

To delete MTA archives with IDs <MTA_ID1> and <MTA_ID2> that have been deployed to your subaccount, execute:

```
neo delete-mta --host hana.ondemand.com --account mysubaccount --user
mymail@example.com --id <ID1>,<ID2>
```

6.3.3.4.40 delete-schema

This command deletes the specified schema, including all data it contains. A schema cannot be deleted if it is still bound to an application. To enforce the deletion, use the force parameter but bear in mind that this will also delete all bindings that still exist.

Schema backups are kept for 14 days and may be used to restore mistakenly deleted data (available by special request only).

```
neo delete-schema -a <subaccount_technical_name> -h <host> -u <e-mail_or_user> -
i <schema_ID>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. Type: URL, for acceptable values see Regions

Required	
`-i, --id`	HANA database or schema ID `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Optional	
`-f, --force`	Forcefully deletes the schema, including all application bindings `Default`: *off* `Type`: switch, takes no value
`--silent`	Suppresses the command line confirmation prompt `Default`: *off* `Type`: switch, takes no value

Example

```
neo delete-schema -a mysubaccount -h hanatrial.ondemand.com -u
mymail@example.com -i myschema
```

Related Information

Administering Database Schemas

6.3.3.4.41 delete-security-rule

This console client command deletes a security group rule configured for a virtual machine.

```
neo delete-security-rule --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --name <vm_name> --source-id <system_ID> --source-
type <source_type> --from-port <from_port> --to-port <to_port>
```

Parameters

Required	
`-a, --account`	Your subaccount technical name. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-n, --name`	Type the name of the virtual machine. `Type`: string `Condition`: Use either `--id` or `--name`
`-i , --id`	ID of the virtual machine `Type`: string `Condition`: Use either `--id` or `--name`
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`--from-port`	The start of the range of allowed ports. The <from_port> value must be less than or equal to the <to_port> value. `Type`: positive integer between 1 and 65535
`--to-port`	The end of the range of allowed ports. The <to_port> value must be greater than or equal to the <from_port> value. `Type`: positive integer between 1 and 65535
`--source-id`	The name of the system that you want to connect from. For a SAP HANA system, the `--source-id` is the SAP HANA database system name. For a Java application, it is the application name.
`--source-type`	The type of the system specified in the source ID. `Acceptable values`: HANA, JAVA

Example

```
neo delete-security-rule --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --name myvm --source-id mysystem --source-type HANA --from-
port <from_port> --to-port <to_port>
```

Related Information

Manage Network Communication for SAP Virtual Machine Service

6.3.3.4.42 delete-vm

Stops and deletes a virtual machine by name or by ID.

```
neo delete-vm --account <subaccount_technical_name> --user <e-mail_or_user> --
host <host> --name <vm_name>
```

> **i** Note
>
> By default, deleting a virtual machine doesn't delete its volume and volume snapshots. This gives you the option to create a new virtual machine from the remaining volume and volume snapshots, and allows you to not lose any data that was installed on the file system. For more information, see Manage Volumes and Manage Volume Snapshots.

However, if you want to delete the virtual machine along with its volume and volume snapshots, you can use the `--delete-volume` and `--delete-volume-snapshots` parameters.

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-i , --id`	ID of the virtual machine `Type`: string `Condition`: Use either `--id` or `--name`
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Required	
`-n, --name`	Name of the virtual machine `Type`: string `Condition`: Use either `--id` or `--name`
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Optional	
`-v, --delete-volume`	Deletes the volume referenced by the virtual machine.
`-s, --delete-volume-snapshots`	Deletes all volume snapshots referenced by the virtual machine.
`-f, --force`	You won't be asked to confirm the deletion of the virtual machine.
`-y, --synchronous`	Waits until the deletion is complete. `Type`: switch; takes no value `Default`: off

Example

```
neo delete-vm --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --name myvm --synchronous
```

6.3.3.4.43 delete-volume

Deletes a specified virtual machine volume.

```
neo delete-volume --id <id_of_the_volume> --account <subaccount_technical_name>
--user <e-mail_or_user> --host <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-v, --id`	Unique identifier of the volume that you want to delete `Type`: string `Condition`: Use either `--id` or `--name`
`-n, --name`	Name of the volume `Type`: string `Condition`: Use either `--id` or `--name`

Optional	
`-s, --delete-volume-snapshots`	Deletes all volume snapshots referenced by the volume.
`-f, --force`	You won't be asked to confirm the deletion of the virtual machine volume.

Example

```
neo delete-volume --id myvolumeid --account mysubaccount --host
hana.ondemand.com --user myemail@example.com
```

6.3.3.4.44 delete-volume-snapshot

Deletes a specified virtual machine volume snapshot.

```
neo delete-volume-snapshot --id <id_of_the_snapshot> --account
<subaccount_technical_name> --user <e-mail_or_user> --host <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-s, --id`	Unique identifier of the volume snapshot that you want to delete `Type`: string `Condition`: Use either `--id` or `--name`
`-n, --name`	Name of the volume snapshot `Type`: string. It can contain only alphanumeric characters (0-9, a-z, A-Z), underscore (_), and hyphen (-). The allowed name length is between 1 and 128 characters. `Condition`: Use either `--id` or `--name`
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Optional	
`-f, --force`	You won't be asked to confirm the deletion of the virtual machine volume snapshot.

Example

```
neo delete-volume-snapshot --id mysnapshotid --account mysubaccount --host
hana.ondemand.com --user myemail@example.com
```

Related Information

Manage Volume Snapshots
create-volume-snapshot [page 1399]
display-volume-snapshot [page 1438]
list-volume-snapshots [page 1512]

6.3.3.4.45 deploy

Deploying an application publishes it to SAP BTP. Use the optional parameters to make some specific configurations of the deployed application.

If you use enhanced disaster recovery, the application is deployed first on the specified region and then on the disaster recovery region.

```
neo deploy --host <host> --account <subaccount_technical_name> --application
<application_name>
--source <file_location> --user <e-mail_or_user>
```

Parameters

To list all parameters available for this command, execute **`neo help deploy`** in the command line.

Required	
`-a`, `--account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required

`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) **i Note** When the sum of the characters of the subaccount technical name and the application name exceeds 30, the default application URL is truncated. This means that parts of the subaccount name and the application name will be missing from the application URL: **Example** Subaccount name: mycompanyextensionspoc Application name: myappsimplesimplereq Application URL: `https://myappsimplemycompanyextensions.us1.hana.ondemand.com`
`-s, --source`	A comma-separated list of file locations, pointing to WAR files, or folders containing them **i Note** The size of an application can be up to 1.5 GB. If the application is packaged as a WAR file, the size of the unzipped content is taken into account. If you want to deploy more than one application on one and the same application process, put all WAR files in the same folder and execute the deployment with this source, or specify them as a comma-separated list. `Type`: file location
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions. To deploy an application in more than one region, execute the deploy separately for each region host.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or username. `Type`: string

Optional

Command-specific parameters

Optional

`--connections`	The number of connections used to deploy an application. Use it to speed up deployment of application archives bigger than 5 MB in slow networks. Choose the optimal number of connections depending on the overall network speed to the cloud. `Default:` *2* `Acceptable values:` **1-6** `Type:` integer
`--delta`	Deploys only the changes between the provided source and the deployed content. New content will be added; missing content will be deleted. Recommended for development use to speed up the deployment. **i Note** The deployment to the disaster recovery region is not supported with this parameter. `Acceptable values:` None
`--ev`	Environment variables for configuring the environment in which the application runs. **i Note** For security reasons, do not specify any confidential information in plain text format, such as usernames and passwords. You can either encrypt such data, or store it in a secure manner. For more information, see Keystore Service [page 1779]. Sets one environment variable by removing the previously set value; can be used multiple times in one execution. `Type:` --ev <KEY1>=<VALUE1> --ev <KEY2>=<VALUE2> , where a key-value pair specifies one environment variable If you provide a key without any value (--ev <KEY1>=), the `-ev` parameter is ignored. For a value that contains spaces, use quotation marks.
`-j, --java-version`	Java Virtual Machine major version number `Default:` depends on the SDK for SAP BTP, Neo environment `Type:` version number of the JRE. For more information, see Choose JRE Version [page 1595].
`-m,`	Minimum number of application processes, on which the application can be started `Default:` *1*
`-M, --maximum-processes`	Maximum number of application processes, on which the application can be started `Default:` *1*

Optional

`-V, --vm-arguments`	Java Virtual Machine arguments System properties (-D<name>=<value>) separated with space that will be used when starting the application process. Memory settings of your compute units. You can set the following memory parameters: -Xms, -Xmx, -XX:PermSize, -XX:MaxPermSize. We recommend that you use the default memory settings. Change them only if necessary and note that this may impact the application performance or its ability to start. For more information, see Configure VM Arguments [page 1598].
`-z, --size`	Compute unit size `Acceptable values`: lite, pro, prem, prem-plus `Default`: the smallest size from the subaccount quotas For more information, see Compute Units [page 867].
`--runtime`	Application runtime Use the parameter if you want to choose an application runtime container different from the one coming with your SDK. To view all available runtime containers, use list-runtimes [page 1500]. For more information, see Application Runtime Container [page 857]. If you specify `--runtime`, you also have to specify `--runtime-version`. **i Note** The deployment with an unsupported runtime will fail with an error message, and the deployment with a deprecated runtime will result in a warning message.

Optional	
`--runtime-version`	The runtime version on which the application will be started and will run on the same version after a restart. Otherwise, by default, the application is started on the latest minor version (of the same major version) which is backward compatible and includes the latest corrections (including security patches), enhancements, and updates. Note that choosing this option does not affect already started application processes. You can view the recommended versions by executing the *list-runtime-versions* command. This is a mapping between the runtime and the respective version: • Java Web corresponds to runtime **neo-java-web** and runtime version **1**. • Java Web Tomcat 7 corresponds to runtime **neo-java-web** and runtime version **2**. • Java Web Tomcat 8 corresponds to runtime **neo-java-web** and runtime version **3**. • Java EE 6 Web Profile corresponds to runtime **neo-javaee6-wp** and runtime version **2**. **i Note** If you choose your runtime version, consider its expiration date and plan updating to a new version regularly. For more information, see Choose Application Runtime Version [page 1594].
Tomcat connector attributes	
`--compression`	Enable or disable gzip response compression `Default`: *off* Possible values: *on* (allow compression), *off* (disable compression), *force* (forces compression for all responses) or an *integer* (which enables compression and specifies the compression-min-size value in bytes). For more information, see Enable and Configure Gzip Response Compression [page 1597]
`--compressible-mime-type`	A comma separated list of MIME types for which compression is used `Default`: *text/html, text/xml, text/plain* `Condition:` applicable if compression is enabled
`--compression-min-size`	Responses bigger than this value get compressed `Condition:` applicable if compression is enabled
`--connection-timeout`	Defines the number of milliseconds to wait for the request URI line to be presented after accepting a connection. `Default`: *20000* **→ Tip** You can use this option to mitigate the threat of slow HTTP attacks by applying the appropriate timeout value for your network setup. See Protection from Web Attacks [page 1835].

Optional	
`--max-threads`	Specifies the maximum number of simultaneous requests that can be handled `Default:` *200*
`--uri-encoding`	Specifies the character encoding used to decode the URI bytes on application request `Default:` *ISO-8859-1* For more information, see the encoding sets supported by Java SE 6 and Java SE 7 .

Example

Here are examples of some additional configurations. If your application is already started, stop it and start it again for the changes to take effect.

You can deploy an application on a host different from the default one by specifying the host parameter. For example, to use the region (host) located in the United States, execute:

```
neo deploy --host us1.hana.ondemand.com --account mysubaccount --application
myapp --source samples/deploy_war/example.war
--user mymail@example.com
```

Choose compute unit size

To specify the compute unit size on which you want the application to run, use the `--size` parameter with one of the following values:

- lite - Lite Edition
- pro - Professional edition
- prem - Premium edition
- prem-plus - Premium Plus edition

Available sizes depend on your subaccount type and what options you have purchased. For trial accounts, only the Lite edition is available.

For more information, see Compute Units [page 867].

For example, if you have an enterprise account and have purchased a package with Premium edition compute units, then you can run your application on a Premium compute unit size, by executing the following command:

```
neo deploy --size prem myapp.properties
```

Set the context root of an application

When deploying an application, name the WAR file with the desired context root.

For example, if you want to deploy your WAR in context root "/hello" then rename your WAR to hello.war.

If you want to deploy it in the "/" context root then rename your WAR to ROOT.war.

Specify character encoding

Using the `--uri-encoding` parameter, you can define the character encoding that will be used to decode the URI bytes on application request. For example, to use the UTF-8 encoding that can represent every character in the Unicode character set, execute

```
neo deploy --uri-encoding UTF-8 myapp.properties
```

Related Information

Console Client for the Neo Environment [page 1348]
Choose Application Runtime Version [page 1594]
Choose JRE Version [page 1595]
Configure VM Arguments [page 1598]
Enable and Configure Gzip Response Compression [page 1597]
Scale Applications [page 1599]
Update Application Properties [page 1593]
Deploying and Updating Applications [page 875]
Use Delta Deployment [page 893]
Org Administration Using the Cockpit
list-applications [page 1481]

6.3.3.4.46 deploy-local

This command deploys WAR files on a local server instance.

```
neo deploy-local --source <file_location>
```

Parameters

Required	
`-s`, `--source`	Source for deployment (comma separated list of WAR files or folders containing one or more WAR files) `Type`: file location

Optional	
`-l`, `--location`	Local server installation directory

Example

```
neo deploy-local --source samples/deploy_war/example.war
```

Related Information

Deploy Locally with the Console Client [page 890]

6.3.3.4.47 deploy-mta

This command deploys Multitarget Application (MTA) archives. One or more than one MTA archives can be deployed to your subaccount in one go.

```
neo deploy-mta --host <host> --account <subaccount_technical_name>  --source
<file(1)_location>,<file(2)_location>,...,<file(n)_location> --user <e-
mail_or_user>
```

Parameters

To list all parameters available for this command, execute **`neo help deploy-mta`** in the command line.

Required	
`-a, --account`	The name of the subaccount for which you provide a user and a password.
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-p, --password`	Your user password. We recommend that you enter it only when prompted, and not explicitly as a parameter in a properties file or the command line.
`-u, --user`	Your user e-mail or SAP ID (SCN) user name.
`-s, --source`	A comma-separated list of file locations, pointing to the MTA archive files, or the folders containing them.

Optional
Command-specific parameters

Optional	
`-y, --synchronous`	Triggers the deployment and waits until the deployment operation finishes. The command without the `--synchronous` parameter triggers deployment and exits immediately without waiting for the operation to finish. Takes no value.
`-e, --extensions`	Defines one or more extensions to the deployment descriptor. A comma-separated list of file locations, pointing to the extension descriptor files, or the folders containing them. For more information, see Defining MTA Extension Descriptors.
`--mode`	Defines whether the deployment method is a standard deployment, or provider deployment. The available values are `import` (default value), or `providerImport`.

Example

You can deploy an MTA archive on a host different from the default one by specifying the host parameter. For example, to use the region (host) located in the United States, execute:

```
neo deploy-mta --host us1.hana.ondemand.com --account mysubaccount --source /usr/
test/deploy_mta/example.mtar --user mymail@example.com
```

Related Information

list-mta-operations [page 1498]

6.3.3.4.48 disable

This command stops the creation of new connections to an application or application process, but keeps the already running sessions alive. You can check if an application or application process has been disabled by executing the status command.

```
neo disable --host <host> --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user>
```

Parameters

To list all parameters available for this command, execute **`neo help disable`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) `Condition`: Not required if using `--application-process-id`
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) `Condition`: Not required if using `--application-process-id`
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-i, --application-process-id`	Unique ID of a single application process. Use it to disable a particular application process instead of the whole application. As the process ID is unique, you do not need to specify subaccount and application parameters. You can list the application process ID by using the `<status>` command. `Default`: none `Type`: string (hexadecimal sequence of 2 to 40 characters)

Example

To disable the whole application, execute:

```
neo disable --host hana.ondemand.com --account mysubaccount --application myapp
--user mymail@example.com
```

To disable a single applcation process, first identify the application process you want to disable by executing **`neo status`**:

```
neo status --host hana.ondemand.com --account mysubaccount --application myapp --
user mymail@example.com
```

From the generated list of application process IDs, copy the ID you need and execute **`neo disable`** for it:

```
neo disable --application-process-id e8df21d  --host hana.ondemand.com --user
mymail@example.com
```

Related Information

Perform Soft Shutdown [page 1615]
start [page 1557]
status [page 1555]
Exit Codes [page 1584]

6.3.3.4.49 display-application-properties

The command displays the set of properties of a deployed application, such as runtime version, minimum and maximum processes, Java version.

```
neo display-application-properties --host <host> --account
<subaccount_technical_name> --application <application_name> --user <e-
mail_or_user>
```

Parameters

To list all parameters available for this command, execute the **`neo help display-application-properties`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

To list the application properties, execute:

```
neo display-application-properties --host hana.ondemand.com --account
mysubaccount --application myapp --user mymail@example.com
```

Related Information

deploy [page 1422]

6.3.3.4.50 display-csr

Returns the certificate signing request (CSR) of the specified certificate.

```
neo display-csr --account <subaccount_technical_name> --user <e-mail_or_user> --
host <host> --name <certificate_name>
--file-name <file-name>
```

If you have several subaccounts in your global account, the `display-csr` command returns all certificate signing requests in these subaccounts.

Parameters

To list all parameters available for this command, execute **`neo help display-csr`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Required	
`-n, --name`	Name of the certificate set to the SSL host Must already be uploaded

Optional	
`-f, --file name`	Name of the local file where the CSR is stored

Example

```
neo display-csr --account mysubaccount --user mymail@example.com --host
hana.ondemand.com
--name myfirstcert --file-name mycsr
```

Related Information

generate-csr [page 1446]
Configuring Custom Domains [page 1649]

6.3.3.4.51 display-ecm-repository

Returns details and settings of one repository, including tenant details.

```
neo display-ecm-repository --account <subaccount_technical_name>  --host <host>
--user <e-mail_or_user> --name <repository_name>
```

Parameters

Required	
`-a, --account`	Subaccount technical name Specify an existing subaccount of which you are already a member. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required	
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-n, --name`	Name of the repository `Type`: string

Optional	
`-t, --tenant`	Tenant alias `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Example

```
neo display-ecm-repository --account acme --host hana.ondemand.com --user
<myemail@example.com> --name ExampleRepository
ExampleRepository
  Display name      : Example Repository
  Description       : This is an example repository with Virus Scan enabled.
  ID                : cdb158efd4212fc00726b035
  Application       : Neo CLI
  Virus Scan        : on
  Tenants           : 1
    ExampleTenant
      Tenant ID     : 39a9c31a-81a5-4c4e-9bd5-8e60681227ad
      Virus Scan    : off
      Content       : 1 GB
      Metadata      : 258 KB
```

6.3.3.4.52 display-db-info

This command displays detailed information about the selected database. This includes the assigned database type, the database version, and a list of bindings with the application and data source names.

```
neo display-db-info -a <subaccount_technical_name> -h <host> -u <e-mail_or_user>
-i <database_ID>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-i, --id`	Database ID `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Example

```
neo display-db-info -a mysubaccount -h hana.ondemand.com -u mymail@example.com -
i mydb
```

6.3.3.4.53 display-mta

This command displays a Multitarget Application (MTA) archive that is deployed to your subaccount.

```
neo display-mta --host <host> --account <subaccount_technical_name> --user <e-
mail_or_user> --id <MTA_ID>
```

Parameters

To list all parameters available for this command, execute **`neo help display-mta`** in the command line.

Required	
`-a, --account`	The name of the subaccount for which you provide a user and a password.

Required	
`-h, --host`	The host on which you execute the command.
`-p, --password`	Your user password. We recommend that you enter it only when prompted, and not explicitly as a parameter in a properties file or the command line.
`-u, --user`	Your user e-mail or SAP ID (SCN) user name.
`-i, --id`	A comma-separated list of MTA IDs

Example

To display an MTA archive with an ID <MTA_ID> that has been deployed to your subaccount, execute:

```
neo display-mta –host hana.ondemand.com --account mysubaccount --user
mymail@example.com --id <ID>
```

6.3.3.4.54 display-schema-info

This command displays detailed information about the selected schema. This includes the assigned database type, the database version, and a list of bindings with the application and data source names.

```
neo display-schema-info -a <subaccount_technical_name> -h <host> -u <e-
mail_or_user> -i <schema_ID>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-i, --id`	Schema ID `Type`: string

Required	
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Example

```
neo display-schema-info -a mysubaccount -h hanatrial.ondemand.com -i myschema -u
mymail@example.com
```

Related Information

Example Scenarios
Administering Database Schemas

6.3.3.4.55 display-volume-snapshot

Shows details about the specified virtual machine volume snapshot.

```
neo display-volume-snapshot --id <snapshot_id> --account
<subaccount_technical_name> --user <e-mail_or_user> --host <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.

Required	
`-s, --id`	Unique identifier of the volume snapshot `Type`: string `Condition`: Use either `--id` or `--name`
`-n, --name`	Name of the volume snapshot `Type`: string. It can contain only alphanumeric characters (0-9, a-z, A-Z), underscore (_), and hyphen (-). The allowed name length is between 1 and 128 characters. `Condition`: Use either `--id` or `--name`
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Example

```
neo display-volume-snapshot --id mysnapshotid --account mysubaccount --host
hana.ondemand.com --user myemail@example.com
```

Related Information

Manage Volume Snapshots
create-volume-snapshot [page 1399]
delete-volume-snapshot [page 1421]
list-volume-snapshots [page 1512]

6.3.3.4.56 download-keystore

This command is used to download a keystore by downloading the keystore file. You can download keystores on subaccount, application, and subscription levels.

Parameters

To list all parameters available for this command, execute **`neo help download-keystore`** in the command line.

Required	
`-a, --account`	Consumer subaccount technical name The subaccount for which you provide username and password. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-n,--name`	Name of the keystore to be downloaded `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-b, --application`	Application name • Use `--application <consumer_application_name>` if the application is running in your subaccount. • Use `--application <provider_subaccount_technical_name>:<provider_application_name>` if the application is running in another subaccount. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-l,--location`	Local directory where the keystore will be saved. If it is not specified, the current directory is used. `Type`: string
`-w, --overwrite`	Overwrites a file with the same name if such already exists. If you do not explicitly include the `--overwrite` argument, you will be notified and asked if you want to overwrite the file.

Example

On Subscription Level

```
neo download-keystore --account <consumer_subaccount_technical_name> --
application <provider_subaccount_technical_name>:<provider_application_name>
--user <e-mail_or_user> --location c:\temp --name KeyStore1 --host
hana.ondemand.com
```

On Application Level

```
neo download-keystore --account <consumer_subaccount_technical_name> --
application <consumer_application_name>
--user <user_ID> --location c:\temp --name KeyStore1 --host hana.ondemand.com
```

On Subaccount Level

```
neo download-keystore --account <consumer_subaccount_technical_name> --user <e-
mail_or_user>
--location c:\temp --name KeyStore1 --host hana.ondemand.com
```

Related Information

Keystore Console Commands [page 1783]
Keys and Certificates [page 1781]
Using the Keystore Service for Client Side HTTPS Connections [page 1791]

6.3.3.4.57 edit-ecm-repository

Changes the name, key, or virus scan settings of a repository. You cannot change the display name or the description.

At least one of the `--newname`, `--newkey`, or `--virus-scan` parameters must be provided.

```
neo edit-ecm-repository --account <subaccount_technical_name>  --host <host> --
user <e-mail_or_user> --name <repository_name> --tenant <tenant_name> --virus-
scan <true/false> --key <repository_key>
```

> **i Note**
>
> With this command, you can also change your current repository key to a different one. If you forgot your current key, request a new one using the `reset-ecm-repository` command.

Parameters

Required	
`-a`, `--account`	Subaccount technical name Specify an existing subaccount of which you are already a member. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required

`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-k, --key`	Key of the repository `Type`: string
`-n, --name`	Name of the repository `Type`: string

Optional

`-t, --tenant`	Tenant alias Changes the virus scan setting for one tenant. ⚠ Caution If not used, the virus scan setting of the whole repository changes. `Type`: string
`-o, --newname`	New name of the repository `Type`: string
`-q, --newkey`	New repository key `Type`: string
`-v, --virus-scan`	Can be used to activate the virus scanner and check all incoming documents for viruses. `Default`: `true` `Type`: boolean → Recommendation For repositories that are used by untrusted users and or for unknown content, we recommend that you enable the virus scanner by setting this parameter to `true`. Enabling the virus scanner could impair the upload performance. If a virus is detected, the upload process for the document fails with a virus scanner exception.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Example

```
neo edit-ecm-repository --account sap --host hana.ondemand.com --user
<myemail@example.com> --name DemoRepository --tenant sap --virus-scan false --
key ecm_012345689
SAP BTP Console Client
edit-ecm-repository executed successfully.
```

Related Information

reset-ecm-key [page 1525]

6.3.3.4.58 enable

This command enables new connection requests to a disabled application or application process. The enable command cannot be used for an application that is in maintenance mode.

```
neo enable --host <host> --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user>
```

Parameters

To list all parameters available for this command, execute **`neo help enable`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) `Condition`: Not required if using `--application-process-id`
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) `Condition`: Not required if using `--application-process-id`
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions

Required	
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-i, --application-process-id`	Unique ID of a single application process. Use it to enable a particular application process instead of the whole application. As the process ID is unique, you do not need to specify subaccount and application parameters. You can list the application process ID by using the `<status>` command. `Default`: none `Type`: string (hexadecimal sequence of 2 to 40 characters)

Example

To enable the whole application, execute:

```
neo enable --host hana.ondemand.com --account mysubaccount --application myapp --
user mymail@example.com
```

To enable a single applcation process, first identify the application process you want to enable by executing **`neo status`**:

```
neo status --host hana.ondemand.com --account mysubaccount --application myapp --
user mymail@example.com
```

From the generated list of application process IDs, copy the ID you need and execute **`neo enable`** for it:

```
neo enable --application-process-id e8df21d  --host hana.ondemand.com --user
mymail@example.com
```

Related Information

status [page 1555]
disable [page 1430]
start-maintenance [page 1560]

6.3.3.4.59 get-destination

This command downloads (reads) destination configuration properties files and destination certificate files. You can download them on subaccount, application or subscribed application level.

```
neo get-destination --account <subaccount_technical_name> --user <e-
mail_or_user> --localpath <localpath_to_destination_file_or_JKS_file> --host
<host>
```

Parameters

To list all parameters available for this command, execute **`neo help get-destination`** in the command line.

Required	
`-a, --account`	Your subaccount. The subaccount for which you provide username and password. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	The application for which you download a destination. Cases: • Use `--application <myapp>` if the application is running in your subaccount. • Use `--application <provider_subaccount>:<provider_app>` if the application is running in another subaccount. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`--localpath`	The path on your local file system where a destination or a JKS file will be downloaded. If not set, no files will be downloaded. `Type`: string
`--name`	The name of the destination or JKS file to be downloaded. If not set, the names of all destination or JKS files for the service will be listed. `Type`: string

Required	
`-p, --password`	Password for the specified user. To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string **i Note** If you download a destination configuration file that contains a password field, the password value will not be visible. Instead, after `Password =...`, you will only see an empty space. You will need to learn the password in other ways.
`-u, --user`	Your email, SAP ID or user name `Type`: string

Examples

- To read a destination on subaccount level, execute:

```
neo get-destination --account mysubaccount --user p1234567890 --name weather
--localpath C:\myfiles --host hanatrial.ondemand.com
```

- To read a destination on application level, execute:

```
neo get-destination --account mysubaccount -user p1234567890 --application
demo --name myconfiguration.jks --localpath C:\SDK\tools\samples\connectivity
--host hanatrial.ondemand.com
```

- To read a destination on subscribed application level, execute:

```
neo get-destination --account mysubaccount --user p1234567890 --application
othersubaccount:remotedemo --name weather --localpath C:\SDK\tools\samples
\connectivity --host hanatrial.ondemand.com
```

Related Information

Download Destinations [page 60]
Exit Codes [page 1584]

6.3.3.4.60 generate-csr

Generates and returns a certificate signing request (CSR).

```
neo generate-csr --account <subaccount_technical_name> --user <e-mail_or_user> --
host <host> --name <certificate_name>
```

```
--certificate-distinguished-name <type0=value0,type1=value1,type2>
```

Parameters

To list all parameters available for this command, execute **`neo help generate-csr`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-n, --name`	Unique identifier of the certificate `Type`: string When generating a CSR, the certificate name must start with a letter and can only contain lowercase letters (a-z), uppercase letters (A-Z), numbers (0-9), underscores (_), and hyphens (-).
`-d,--certificate-distinguished-name`	Attributes of the CSR `Type`: string (formatted as type0=value0,type1=value1,type2=..., characters may be escaped by \ (backslash), no spaces are skipped) `Allowed attributes`: • Country (C) – two-digit code - for example, 'GB' • State (ST) – state or province name - for example, 'Hampshire' • Locality (L) – city full name - for example, 'Portsmouth' • Organization (O) – company name • Organizational Unit (OU) – for example, 'IT Department' • Common Name (CN) – the domain name for which you are requesting the certificate - '`www.example.com`' • Email Address (E) – to validate the certificate request, some certificate authorities (CA) require the email address of the domain owner

Optional	
`-s, --subject-alternative-name`	A comma-separated list of all domain names to be protected with this certificate, used as value for the Subject Alternative Name field of the generated certificate. `Type`: string
`-g, --signature-algorithm`	Signature algorithm `Default`: SHA256withRSA `Supported algorithms`: SHA256withRSA, SHA384withRSA, SHA512withRSA
`-k, --key-size`	The size (length) of a generated key pair `Supported sizes`: 2048, 4096

Example

```
neo generate-csr --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --name myfirstcert
--certificate-distinguished-name
"C=BG,O=MyCompany,CN=www.mycompany.bg,E=admin@mycompany.bg" --subject-
alternative-name “www.mycompany.com,www.mycompany.net”
```

Related Information

Configuring Custom Domains [page 1649]
list-domain-certificates [page 1489]
delete-domain-certificate [page 1408]

6.3.3.4.61 get-log

This command downloads a particular log file.

```
neo get-log --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user>  --host <host> --directory
<local_folder_location_of_the_file> --file <file_name>
```

Parameters

To list all parameters available for this command, execute **`neo help get-log`** in the command line.

Required

`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-d, --directory`	Local folder location under which the file will be downloaded. If the directory you have specified does not exist, it will be created. `Type`: string
`-f, --file`	The log file name including its extension. `Type`: string **i Note** To find out the name of the log file to download, use the `list-logs` command to see the available log files of your application. For more information, see list-logs [page 1496].
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-p, --password`	Password for the specified user. To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Your email, SAP ID or user name `Type`: string

Optional

`-w, --overwrite`	Overwrites a file with the same name if such already exists. If you do not explicitly include the `--overwrite` argument, you will be notified and asked if you want to overwrite the file. `Default`: true `Type`: boolean

Optional	
`-t`, `--apptype`	Application type. Use this parameter to differentiate applications with the same name from different technologies (Java and HTML5). The following values are allowed for the application type: • `java` - for Java applications • `html5` - for HTML5 applications `Default`: `java` `Type`: string

Example

```
neo get-log --account mysubaccount --application demo --apptype html5 --user
p1234567890 --host hana.ondemand.com --directory C:\MyDemoApps\log --file
ljs_trace_2016-08-25.log
```

Related Information

Using Logs in the Console Client
Exit Codes [page 1584]

6.3.3.4.62 grant-db-access

This command gives another subaccount permission to access a database. The subaccount providing the permission and the subaccount receiving the permission must be part of the same global account.

```
neo grant-db-access --account <subaccount_ID> --user <e-mail_or_user> --host
<host> --id <database_ID>
--to-account <to-subaccount_technical_name> --permissions <permission_type>
```

Parameters

Required	
`-a, --account`	ID of the subaccount providing the database. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions

Optional	
`-i, --id`	ID of the database to which access will be given.
`-to-account`	The subaccount to receive access permission. The subaccount provoding the permission and the subaccount receiving the permission must be part of the same global account.
`-permissions`	Comma-separated list of access permissions to the database. Acceptable values: 'TUNNEL', 'BINDING'.

Example

```
neo grant-db-access --account myProviderSubaccount --user mymail@example.com --
host hana.ondemand.com --id myDB --to-account myConsumerSubaccount --permissions
TUNNEL, BINDING
```

Related Information

list-db-access-permissions [page 1485]
revoke-db-access [page 1531]
Managing Access Permissions
Sharing Databases in the Same Global Account

6.3.3.4.63 grant-db-tunnel-access

This command generates a token, which allows the members of another subaccount to access a database using a database tunnel.

```
neo grant-db-tunnel-access -h <host> -u <user> -a <my subaccount> -i
<mydatabase> --to-account <other subaccount>
```

Parameters

Required	
`-i, --id`	Database ID `Type`: string
`--to-account`	Subaccount technical name The subaccount to be granted database tunnel access, based on the access token `Type`: string

Example

```
neo grant-db-tunnel-access -h hanatrial.ondemand.com -u mymail@example.com -a
mysubaccount -i mydb --to-account other subaccount
```

Related Information

Give Other Subaccounts Permission to Open a Database Tunnel
list-db-tunnel-access-grants [page 1490]
revoke-db-tunnel-access [page 1532]

6.3.3.4.64 grant-schema-access

This command gives an application in another subaccount access to a schema based on a one-time access token. The access token is used to bind the schema to the application.

```
neo grant-schema-access --host <host> --account <subaccount_technical_name> --
application <application_name> --user <e-mail_or_user> --id <schema_ID>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name The application (specified in the format <subaccount>:<application>) to which the schema can be bound using the created token `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-i, --id`	Schema ID `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Example

```
neo grant-schema-access --host hanatrial.ondemand.com --account mysubaccount --
application salescorp:salesapp --user mymail@example.com --id myschema
```

Related Information

list-schema-access-grants [page 1503]
revoke-schema-access [page 1533]
Managing Access to Databases for Other Subaccounts
Sharing Databases with Other Subaccounts
Grant Access to Schemas
bind-schema [page 1370]

6.3.3.4.65 hcmcloud-create-connection

Use this command to configure the connectivity of an extension application to an SAP SuccessFactors system associated with a specified subaccount in the Neo environment, or to configure the connectivity of a specified subaccount in the Neo environment to an SAP SuccessFactors system associated with this subaccount. The command creates the required HTTP destination and registers an OAuth client for the extension application in SAP SuccessFactors.

Prerequisites

To be able to use this command, you need the following platform scopes to be specified for your custom platform role:

- readExtensionConfigurations
- manageExtensionConfigurations
- manageDestinations

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

```
neo  hcmcloud-create-connection --application <Java_extension_application> --
account <subaccount_technical_name> --user <e_mail or user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help hcmcloud-create-connection`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions and Hosts.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name. `Type`: string

Optional

`-w, --overwrite`	If a connection with the same name already exists, overwrites it. If you do not explicitly specify the --overwrite parameter, and a connection with the same name already exists, the command fails to execute
`--technical-user-id`	ID of the technical user for the connection. `Condition`: Required for connection type OData with technical user `Type`: string
`-b, --application`	The name of the extension application for which you are creating the connection. Cases: • Use `--application <my_extension_application>` if the application is running in your subaccount • Use `--application <provider_subaccount>:<extension_application application>` if the application is running in another subaccount and your extension subaccount is subscribed to the application. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) **i Note** This parameter is only relevant for Java applications.
`-n, --name`	Name of the connection. `Default`: • For a connection of OData type: `<sap_hcmcloud_core_odata>` • For a connection of OData with a technical user type: `<sap_hcmcloud_core_odata_technical_user>` `Condition`: If you have not specified the name parameter, the default value `<sap_hcmcloud_core_odata>` will be assumed for connections without a technical user, the default value `<sap_hcmcloud_core_odata_technical_user>` will be assumed for connections with technical user. **i Note** If the connection is on a subaccount level, the name is required and must be different than `<sap_hcmcloud_core_odata>` and `<sap_hcmcloud_core_odata_technical_user>`. `Type`: string (up to 200 characters; uppercase and lowercase letters, numbers, and the special characters en dash (-) and underscore (_).

Example

To configure a connection of type OData with technical user and with a specific name for a Java extension application in a subaccount located in the United States (US East) region, execute:

```
neo hcmcloud-create-connection --application <my_Java_application> --account
<my_extension_subaccount> --user <my_email@example.com> --host
us1.hana.ondemand.com --technical-user-id <technical_user_id> --name
<my_connection_name>
```

Result

After executing this command without specifying a name for the connection, you have one of the following destinations in your subaccount:

- sap_hcmcloud_core_odata
- sap_hcmcloud_core_odata_technical_user

After executing the command with a specific name for the connection, the required destination is created in your subaccount.

You can consume this destination in your application using one of these APIs:

- HttpDestination API and DestinationFactory
- ConnectivityConfiguration API
- AuthenticationHeaderProvider API

6.3.3.4.66 hcmcloud-delete-connection

This command removes the specified connection configured between an extension application and a SAP SuccessFactors system associated with the specified subaccount in the Neo environment, or between a specified subaccount in the Neo environment and the SAP SuccessFactors system associated with it.

Prerequisites

To be able to use this command, you need the following platform scopes to be specified for your custom platform role:

- readExtensionConfigurations
- manageExtensionConfigurations
- manageDestinations

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

```
neo hcmcloud-delete-connection --application <extension_application> --account
<subaccount_technical_name> --user <e_mail or user> --host <host> --name
<destination_name>
```

Parameters

To list all parameters available for this command, execute **`neo help hcmcloud-delete-connection`** in the command line.

Required	
`-n, --name`	Name of the connection destination `Type`: string (up to 200 characters; uppercase and lowercase letters, numbers, and the special characters en dash (-) and underscore (_).
`-a, --account`	Subaccount technical name `Type`: string (up to 200 characters; uppercase and lowercase letters, numbers, and the special characters en dash (-) and underscore (_).
`-h, --host`	Enter a region host `Type`: URL. For acceptable values see Regions and Hosts.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Example

To delete an OData connection for an extension application running in an extension subaccount in the US East region, execute:

```
neo hcmcloud-delete-connection --application <extension_application> --account
<subaccount_technical_name> --user <e_mail or user> --host us1.hana.ondemand.com
--name sap_hcmcloud_core_odata
```

6.3.3.4.67 hcmcloud-disable-application-access

This command removes an extension application from the list of authorized assertion consumer services for the SAP SuccessFactors system associated with the specified subaccount.

Prerequisites

To be able to use this command, you need the following platform scopes to be specified for your custom platform role:

- readExtensionConfigurations
- manageExtensionConfigurations

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

```
neo hcmcloud-disable-application-access --application <extension_application> --
application-type <extension_application_type> --account
<extension_subaccount_technical_name> --user <e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help hcmcloud-disable-application-access`** in the command line.

Required	
`-b, --application`	The name of the extension application for which you are deleting the connection. Cases: • Use `--application <my_extension_application>` if the application is running in your subaccount • Use `--application <provider_subaccount>:<extension_application>` if the application is running in another subaccount and your extension subaccount is subscribed to the application `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`--application-type`	The type of the extension application for which you are deleting the connection `Accepted values`: **`java`**, **`html5`** `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host `Type`: URL. For acceptable values see Regions and Hosts.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

To remove a Java extension application from the list of authorized assertion consumer services for the SAP SuccessFactors system associated with a subaccount located in the United States (US East), execute:

```
neo  hcmcloud-disable-application-access --application <my_application> --
application-type java --account <my_extension_subaccount>  --user
<my_email@example.com> --host us1.hana.ondemand.com
```

The command removes the entry for the application from the list of the authorized service provider assertion consumer services for the SAP SuccessFactors system associated with the specified subaccount. If entry for the extension application does not exist the command will fail.

6.3.3.4.68 hcmcloud-display-application-access-status

This command displays the status of an extension application entry in the list of assertion consumer services for the SAP SuccessFactors system associated with the specified subaccount. The returned results contain the extension application URL.

Prerequisites

To be able to use this command, you need the following platform scopes to be specified for your custom platform role:

- readExtensionConfigurations
- readHTML5Applications

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

```
neo hcmcloud-display-application-access-status --application
<extension_application> --account <extension_subaccount_technical_name> --user
<e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help hcmcloud-display-application-access-status`** in the command line.

Required	
`-b, --application`	The name of the extension application for which you are displaying the status in in the list of assertion consumer services. Cases: • Use `--application <my_extension_application>` if the application is running in your subaccount • Use `--application <provider_subaccount>:<extension_application>` if the application is running in another subaccount and your extension subaccount is subscribed to the application `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`--application-type`	The type of the extension application for which you are creating the connection `Accepted values`: **`java`**, **`html5`** `Type`: string
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host `Type`: URL. For acceptable values see Regions and Hosts.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

To display the status of an application entry in the list of authorized assertion consumer services for the SAP SuccessFactors system associated with a subaccount in the region located in the United States (US East), execute:

```
neo  hcmcloud-display-application-access-status --application myapp --account
myextensionsubacc  --user mymail@example.com --host us1.hana.ondemand.com
```

6.3.3.4.69 hcmcloud-enable-application-access

This command registers an extension application as an authorized assertion consumer service for the SAP SuccessFactors system associated with the specified subaccount to enable the application to use the SAP SuccessFactors identity provider (IdP) for authentication.

Prerequisites

To be able to use this command, you need the following platform scopes to be specified for your custom platform role:

- readExtensionConfigurations
- manageExtensionConfigurations
- readHTML5Applications

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

```
neo hcmcloud-enable-application-access --application <extension_application> --
account <subaccount_technical_name> --user <e-mail_or_user> --host <host> --
application-type <extension_application_type>
```

Parameters

To list all parameters available for this command, execute **`neo help hcmcloud-enable-application-access`** in the command line.

Required	
`-b, --application`	The name of the extension application for which you are creating the connection. Cases: • Use `--application <my_extension_application>` if the application is running in your subaccount • Use `--application <provider_subaccount>:<extension_application>` if the application is running in another subaccount and your extension subaccount is subscribed to the application `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`--application-type`	The type of the extension application for which you are creating the connection `Accepted values`: **`java`**, **`html5`** `Type`: string
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host `Type`: URL. For acceptable values see Regions and Hosts.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

To register an extension application as an authorized assertion consumer service for the SAP SuccessFactors system associated with a subaccount located in the United States (US East) region, execute:

```
neo  hcmcloud-enable-application-access --application <my_application> --account
<my_extension_subaccount>  --user <my_email@example.com> --host
us1.hana.ondemand.com --application-type java
```

The command creates entry for the application in the list of the authorized service provider assertion consumer services for the SAP SuccessFactors system associated with the specified subaccount. The entry contains the main URL of the extension application, the service provider audience URL and service provider logout URL. If an entry for the given extension application already exists, this entry is overwritten.

6.3.3.4.70 hcmcloud-enable-role-provider

This command enables the SAP SuccessFactors role provider for the specified Java application.

Prerequisites

To be able to use this command, you need the following platform scopes to be specified for your custom platform role:

- readExtensionConfigurations
- manageExtensionConfigurations
- readDestinations

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

```
neo  hcmcloud-enable-role-provider --application <extension_application> --
account <subaccount_technical_name> --user <e_mail or user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help hcmcloud-enable-role-provider`** in the command line.

Required	
`-b, --application`	The name of the extension application for which you are creating the connection. Cases: • Use `--application <my_extension_application>` if the application is running in your subaccount • Use `--application <provider_subaccount>:<extension_application>` if the application is running in another subaccount and your extension subaccount is subscribed to the application `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host `Type`: URL. For acceptable values, see Regions and Hosts.

Required	
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`--connection-name`	The name of the destination for connecting to the SAP SuccessFactors system OData API. `Default`: `<sap_hcmcloud_core_odata>` `Condition`: If you do not specify a value for the `--connection-name` parameter, the default value will be used. `Type`: string (up to 200 characters; uppercase and lowercase letters, numbers, and the special characters en dash (-) and underscore (_).

Example

To enable the SAP SuccessFactors role provider for your Java application in an extension subaccount located in the United States (US East) region, execute:

```
neo hcmcloud-enable-role-provider --application <my_application> --account
<my_extension_subaccount> --user <my_email@example.com> --host
us1.hana.ondemand.com
```

6.3.3.4.71 hcmcloud-get-registered-home-page-tiles

This command lists the SAP SuccessFactors Employee Central (EC) home page tiles registered in the SAP SuccessFactorss company instance associated with the extension subaccount.

Prerequisites

To be able to use this command, you need the following platform scopes to be specified for your custom platform role:

- readExtensionConfigurations
- readHTML5Applications

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

```
neo hcmcloud-get-registered-home-page-tiles --application
<extension_application> --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --application-type <extension_application_type>
```

Parameters

To list all parameters available for this command, execute **`neo help hcmcloud-get-registered-home-page-tiles`** in the command line.

Required	
`-b, --application`	The name of the extension application for which you are listing the home page tiles. Cases: • Use `--application <my_extension_application>` if the application is running in your subaccount • Use `--application <provider_subaccount>:<extension_application>` if the application is running in another subaccount and your extension subaccount is subscribed to the application `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) **i Note** If you do not specify the `application` parameter, the command lists all tiles registered in the Successfactors company instance associated with the specified extension subaccount.
`--application-type`	The type of the extension application for which you are listing the home page tiles `Accepted values`: **`java`**, **`html5`** `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-a, --account`	Name of the extension subaccount `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host `Type`: URL. For acceptable values see Regions and Hosts.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Required	
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

To list the home page tiles registered for a Java extension application running in your subaccount in the US East region, execute:

```
neo  hcmcloud-get-registered-home-page-tiles --application <my_application> --
account <my_extension_subaccount>  --user <my_email@example.com> --host
us1.hana.ondemand.com --application-type <java>
```

There is no lifecycle dependency between the tiles and the application, so the application may not be started or may not be deployed anymore.

6.3.3.4.72 hcmcloud-import-roles

This command imports SAP SuccessFactors HXM suite roles into the SAP SuccessFactors customer instance linked to an extension subaccount.

Prerequisites

To be able to use this command, you need the following platform scopes to be specified for your custom platform role:

- readExtensionConfigurations
- manageExtensionConfigurations

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

```
neo hcmcloud-import-roles --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --location <path_to_the_file_with_role_definitions>
```

Parameters

To list all parameters available for this command, execute **`neo help hcmcloud-import-roles`** in the command line.

Required	
`-l, --location`	Path to the file containing in which the SAP SuccessFactors HXM Suite roles are defined `Type`: string **i Note** The file size must not exceed 500 KB.
`-a, --account`	The extension subaccount in SAP BTP which is linked to the target SAP SuccessFactors system `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host `Type`: URL. For acceptable values see Regions and Hosts.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

To import the role definitions for an extension application from the system repository for your extension subaccount into the SAP SuccessFactors customer instance connected to this subaccount, execute:

```
neo hcmcloud-import-roles --account myextensionsubacc --user mymail@example.com
--host hana.ondemand.com --location pathtorolefile
```

If any of the roles that you are importing already exists in the target system, the commands fails to execute.

Related Information

Create the Resource File with Role Definitions

6.3.3.4.73 hcmcloud-list-connections

This command lists the connections configured for a specified extension application or for a specified subaccount.

Prerequisites

To be able to use this command, you need the following platform scopes to be specified for your custom platform role:

- readExtensionConfigurations
- readDestinations

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

```
neo hcmcloud-list-connections --application <extension_application> --account
<subaccount_technical_name> --user <e_mail or user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help hcmcloud-list-connections`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions and Hosts.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Example

To list the connections for an extension application running in an extension subaccount in the US East region, execute:

```
neo  hcmcloud-list-connections --application myapp --account myextensionsubacc
--user mymail@example.com --us1.host hana.ondemand.com
```

6.3.3.4.74 hcmcloud-register-home-page-tiles

This command registers the SAP SuccessFactors Employee Central (EC) home page tiles in the SAP SuccessFactors company instance associated with the extension subaccount. The home page tiles must be described in a tile descriptor file for the extension application in JSON format.

Prerequisites

To be able to use this command, you need the following platform scopes to be specified for your custom platform role:

- readExtensionConfigurations
- manageExtensionConfigurations
- readHTML5Applications

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

```
neo hcmcloud-register-home-page-tiles --application <extension_application> --
account <subaccount_technical_name> --user <e-mail_or_user> --host <host> --
application-type <extension_application_type> --location
<path_to_the_tile_descriptor_file>
```

Parameters

To list all parameters available for this command, execute **`neo help hcmcloud-register-home-page-tiles`** in the command line.

Required	
`-l, --location`	Path to the tile descriptor file `Type`: string **i Note** The file size must not exceed 100 KB.
`-b, --application`	The name of the extension application for which you are registering the home page tiles. Cases: • Use `--application <my_extension_application>` if the application is running in your subaccount • Use `--application <provider_subaccount>:<extension_application>` if the application is running in another subaccount and your extension subaccount is subscribed to the application `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`--application-type`	The type of the extension application for which you are registering the home page tiles `Default`: **`java`** `Accepted values`: **`java`**, **`html5`** `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-a, --account`	Name of the extension subaccount `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions and Hosts.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

To register a home page tile for a Java extension application running in your subaccount in the US East region, execute::

```
neo  hcmcloud-register-home-page-tiles --application <my_application> --account
<my_extension_subaccount>  --user <my_email@example.com> --host
```

```
us1.hana.ondemand.com --application-type <java> --location
<path_to_tile_descriptor_file>
```

Related Information

Home Page Tiles JSON File

6.3.3.4.75 hcmcloud-unregister-home-page-tiles

This command removes the SAP SuccessFactors EC home page tiles registered for the extension application in the SAP SuccessFactors company instance associated with the specified extension subaccount.

Prerequisites

To be able to use this command, you need the following platform scopes to be specified for your custom platform role:

- readExtensionConfigurations
- manageExtensionConfigurations

For more information, see Platform Scopes and Manage Custom Platform Roles in the Neo Environment.

```
neo hcmcloud-unregister-home-page-tiles --application <extension_application> --
account <subaccount_technical_name> --user <e-mail_or_user> --host <host> --
application-type <extension_application_type>
```

Parameters

To list all parameters available for this command, execute **`neo help hcmcloud-unregister-home-page-tiles`** in the command line.

Required	
`-b, --application`	The name of the extension application for which you are removing the home page tiles. Cases: • Use `--application <my_extension_application>` if the application is running in your subaccount • Use `--application <provider_subaccount>:<extension_application>` if the application is running in another subaccount and your extension subaccount is subscribed to the application `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) **i Note** You must use the same application name that you have specified when registering the tiles.
`--application-type`	The type of the extension application for which you are listing the home page tiles `Default`: **`java`** `Accepted values`: **`java`**, **`html5`** `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-a, --account`	Name of the extension subaccount `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions and Hosts.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

To remove the home page tiles registered for a Java extension application running in your subaccount in the US East region, execute:

```
neo  hcmcloud-unregister-home-page-tiles --application <my_application> --
account <my_extension_subaccount>  --user <my_email@example.com> --host
us1.hana.ondemand.com --application-type <java>
```

There is no lifecycle dependency between the tiles and the application, so the application may not be started or may not be deployed anymore.

6.3.3.4.76 hot-update

The hot-update command enables a developer to redeploy and update the binaries of an application started on one process faster than the normal deploy and restart. Use it to apply and activate your changes during development and not for updating productive applications.

There are three options for hot-update specified with the *--strategy* parameter:

- replace-binaries - redeploys and updates the application binaries
- restart-runtime - redeploys and updates the application binaries and restarts the application process
- reprovision-runtime - cleans up the file system, reprovisions the runtime and redeploys and updates the application binaries

```
neo hot-update --host <host> --account <subaccount_technical_name> --application
<application_name> --source <file_location> --user <e-mail_or_user> --strategy
<update_strategy>
```

Limitations:

- Works only if there is a single running process of the application.
- You cannot change deploy parameters and context path of the application.
- This command does not support enhanced disaster recovery.

Parameters

To list all parameters available for this command, execute **`neo help hot-update`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.

Required	
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Your email, SAP ID or user name. `Type`: string
`-s, --source`	A comma-separated list of file locations, pointing to WAR files, or folders containing them. `Type`: file location
`--strategy`	Defines how the update will be performed. `Acceptable values:` • **`replace-binaries`** • **`restart-runtime`** • **`reprovision-runtime`**

Optional	
`--connections`	Number of connections used to deploy the content `Default:` *`1`* `Acceptable values:` **`1-6`** `Type`: integer
`--delta`	Uploads only the changes between the provided source and the deployed content. New content will be added; missing content will be deleted. Recommended for development use to speed up the deployment. `Acceptable values`: None
`-y, --synchronous`	Waits for the operation to complete.

Example

```
neo hot-update --host us1.hana.ondemand.com --account mysubaccount --application
myapp --source samples/deploy_war/example.war --user mymail@example.com --
strategy replace-binaries
```

6.3.3.4.77 install-local

This command installs a server runtime in a local folder, by default `<SDK installation folder>/server`.

```
neo install-local
```

Parameters

Optional	
`--ajp-port`	AJP port opened by server `Default:` *8009*
`--http-non-proxy-hosts`	JVM system property `http.nonProxyHosts`
`--http-port`	HTTP port opened by server `Default:` *8080*
`--http-proxy-host`	JVM system property `http.ProxyHost`
`--http-proxy-port`	JVM system property `http.ProxyPort`
`--https-port`	HTTPS port opened by server `Default:` *8443*
`--https-proxy-host`	JVM system property `https.ProxyHost`
`--https-proxy-port`	JVM system property `https.ProxyPort`
`--jmx-port`	JMX port opened by server (JVM system property `com.sun.management.jmxremote.port`) `Default:` *1717*
`-l, --location`	Local server installation directory

Related Information

Deploy Locally with the Console Client [page 890]

6.3.3.4.78 list-application-datasources

This command lists all schemas and productive database instances bound to an application.

```
neo list-application-datasources -a <subaccount_technical_name> -b <application>
-h <host> -u <e-mail_or_user>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letters)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Example

```
neo list-application-datasources -a mysubaccount -b myapp -h hana.ondemand.com -
u mymail@example.com
```

Related Information

bind-schema [page 1370]
unbind-schema [page 1572]
bind-hana-dbms [page 1368]
unbind-hana-dbms [page 1571]
Example Scenarios
Administering Database Schemas

6.3.3.4.79 list-availability-check

Lists the availability checks.

```
neo list-availability-check -a <subaccount_technical_name> -u <e-mail_or_user> -
h <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions

Optional	
`-b, --application`	Application name for Java or HTML5 applications, or productive SAP HANA instance database name and application name in the format `<instance name>:<application name>` for SAP HANA XS applications `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-R, --recursively`	Lists availability checks recursively starting from the specified level. For example, if only 'account' is passed as an argument, it starts from the subaccount level and then lists all checks configured on application level. `Default`: false `Type`: boolean

Example

Example for listing availability checks recursively starting on subaccount level and listing the checks configured for Java and SAP HANA XS applications:

```
neo list-availability-check -a mysubaccount -u p1234567 --host hana.ondemand.com
-R
```

Sample output:

```
 Neo Console Client
Running list-availability-checks with the following parameters:
    account    : mysubaccount
    host       : https://hana.ondemand.com
    recursively: true
    SDK version: 1.2.3
    user       : p1234567
HANA XS Availability Checks
Application availability checks
    application : hanaxs:myhana
    url         : /myhana.xsjs
    warning     : 50 s
    critical    : 60 s
    technology  : HANA XS
Java Availability Checks
Subaccount-level availability check
    account     : test
    url         : /example
    warning     : 50
    critical    : 60
Application availability checks
            application : demo
            url         : /example
            warning     : 6
            critical    : 4
            technology  : Java
```

Related Information

Monitoring Java Applications [page 729]
Monitoring Database Systems [page 760]

6.3.3.4.80 list-accounts

Lists all subaccounts that a customer has. Authorization is performed against the subaccount passed as `--account` parameter.

```
neo list-accounts --account <subaccount_technical_name> --user <e-mail_or_user>
--host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help list-accounts`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.

Example

```
neo list-accounts --account mysubaccount --user myuser --host hana.ondemand.com
```

6.3.3.4.81 list-alert-recipients

Lists alert recipients.

Prerequisites

- You've set up the console client.
  For more information, see Set Up the Console Client [page 847].
- The *readMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

```
neo list-alert-recipients -a <subaccount_technical_name> -u <e-mail_or_user> -h
<host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID, or user name `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.

Optional	
`-b, --application`	Application name for Java or HTML5 applications, or productive SAP HANA instance database name and application name in the format `<instance name>:<application name>` for SAP HANA XS applications `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-R, --recursively`	Lists alerts recipients recursively starting from the specified level. For example, if only 'subaccount' is passed as an argument, it starts from the subaccount level and then lists all recipients configured on application level. `Default`: false `Type`: boolean

Example

```
neo list-alert-recipients -a mysubaccount -b demo -u p1234567 -R --host
hana.ondemand.com
```

Sample output:

```
Neo Console Client
Password for your user:
Running list-alert-recipients with the following parameters:
    account     : mysubaccount
    host        : https://hana.ondemand.com
    recursively : true
    user        : p1234567
Subaccount-level alert recipients
Recipients not set on subaccount level for subaccount mysubaccount

       application : demo1
       alert_recipients@example.com
       application : demo2
       alert_recipients@example.org, alert_recipients@example.net
```

Related Information

Monitoring Java Applications [page 729]
Monitoring Database Systems [page 760]
set-alert-recipients [page 1538]
clear-alert-recipients [page 1373]

6.3.3.4.82 list-applications

Lists all Java applications in the specified subaccount.

```
neo list-applications --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Example

```
neo list-applications --account mysubaccount --user mymail@example.com --host
hana.ondemand.com
```

Related Information

deploy [page 1422]
undeploy [page 1574]

6.3.3.4.83 list-application-domains

Lists all domain names on which an application can be accessed.

```
neo list-application-domains --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help list-application-domains`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

```
neo list-application-domains --account mysubaccount --application myapplication
--user mymail@example.com --host hana.ondemand.com
```

Related Information

Add the Custom Domain [page 1655]
add-custom-domain [page 1361]
Configuring Custom Domains [page 1649]

6.3.3.4.84 list-cas

Lists trusted CA certificates in a bundle or bundles that are assigned to an SSL host or hosts.

```
neo list-cas --account <subaccount_technical_name> --host <host> --user <e-mail_or_user>
```

If you have several subaccounts in your global account and you don't list a concrete bundle, the command returns all bundles in these subaccounts.

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name. `Type`: string

Optional	
`--bundle`	Name of the CA bundle that you want to view. `Type`: string You use this parameter instead of `--all`.

Optional	
`--all`	Lists the names of all bundles in the subaccount. Takes no value. You use this parameter instead of `--bundle`.
`-f, --file-name`	Path to the file where the CA bundle is downloaded. `Type`: string `Default`: The CA certificates are saved in the current folder in a file named after the CA bundle. **i Note** If a file with the same name already exists in the specified directory, you will be asked if you want to overwrite the file.

Example

If you want to list a specific bundle in a subaccount, execute:

```
neo list-cas --account mysubaccount --host hana.ondemand.com --user
mymail@example.com --bundle mybundle
```

If you want to list all bundles in a subaccount, execute:

```
neo list-cas --account mysubaccount --host hana.ondemand.com --user
mymail@example.com --all
```

Related Information

add-ca [page 1360]
remove-ca [page 1522]
set-ssl-host [page 1552]
Managing Client Certificate Authentication for Custom Domains [page 1660]

6.3.3.4.85 list-custom-domain-mappings

Lists custom domains configured as access points for applications in a subaccount.

```
neo list-custom-domain-mappings --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host>
```

If you have several subaccounts in your global account, the `list-custom-domain-mappings` command returns all access points in these subaccounts.

Parameters

To list all parameters available for this command, execute **`neo help list-custom-domain-mappings`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

```
neo list-custom-domain-mappings --account mysubaccount --user mymail@example.com
--host hana.ondemand.com
```

Related Information

Configuring Custom Domains [page 1649]

6.3.3.4.86 list-db-access-permissions

This command lists the permissions that other subaccounts have for accessing databases in the specified subaccount.

```
neo list-db-access-permissions --account <subaccount_ID> --user <e-mail_or_user>
--host <host> --id <database_ID>
```

```
--to-account <to-subaccount_technical_name> --permissions <permission_type>
```

Parameters

Required	
`-a, --account`	ID of the subaccount providing the database. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.

Optional	
`-i, --id`	Specify a database to view the permissions only to that database.
`-to-account`	Specify an subaccount to view the permissions only for that subaccount.
`-permissions`	Filter the result by permission. Acceptable values: comma separated list of 'TUNNEL', 'BINDING'.

Example

```
neo list-db-access-permissions --account myProviderSubaccount --user
mymail@example.com --host hana.ondemand.com --permissions TUNNEL, BINDING
```

Related Information

revoke-db-access [page 1531]
grant-db-access [page 1450]
Sharing Databases in the Same Global Account
Managing Access Permissions

6.3.3.4.87 list-dbms

This command lists the dedicated and shared database management systems available for the specified subaccount with the following details: database system (for dedicated databases), database type, and database version.

```
neo list-dbms -a <subaccount_technical_name> -h <host> -u <e-mail_or_user>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Example

```
neo list-dbms -a mysubaccount -h hanatrial.ondemand.com -u mymail@example.com
```

Related Information

Example Scenarios
Administering Database Schemas

6.3.3.4.88 list-dbs

This command lists all databases for the specified subaccount.

```
neo list-dbs -a <subaccount_technical_name> -h <host> -u <e-mail_or_user>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Optional	
`--verbose`	Displays additional information about each database: database type and database version `Default`: *off* `Type`: switch, takes no value

Example

```
neo list-dbs -a mysubaccount -h hana.ondemand.com -u mymail@example.com
```

6.3.3.4.89 list-domain-certificates

Use this command to list certificates available for a custom domain.

```
neo list-domain-certificates --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host>
```

If you have several subaccounts in your global account, the `list-domain-certificates` command returns all certificates in these subaccounts.

Parameters

To list all parameters available for this command, execute **`neo help list-domain-certificates`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

```
neo list-domain-certificates --account mysubaccount --user mymail@example.com --
host hana.ondemand.com
```

Related Information

upload-domain-certificate [page 1578]
delete-domain-certificate [page 1408]
Bind the Certificate to the SSL Host [page 1654]

6.3.3.4.90 list-db-tunnel-access-grants

This command lists all current database access permissions for databases in other subaccounts.

```
neo list-db-tunnel-access-grants -h <host> -u <user> -a <my subaccount>
```

> **i** Note
>
> The list does not include access permissions that have been revoked.

Parameters

Optional	
`-i, --id`	(Optional) Database ID Lists only the access permissions for the specified database `Type`: string

Example

```
neo list-db-tunnel-access-grants -a mysubaccount -h hanatrial.ondemand.com -u
mymail@example.com -i mydb
```

The table below shows the currently active database tunnel access permissions:

Database ID	Granted To	Access Token
myownhana	acmecorp	31t0dpim6rtxa00wx5483vqe7in8i3c1p hv759w9oqrutf638l
myotherhana	acmetest	vm6431dhjcr2e3dbt0fk6jpzm2w7oo3q 48yumf1c6uu8b9pt9z

Related Information

revoke-db-tunnel-access [page 1532]
grant-db-tunnel-access [page 1452]
Give Other Subaccounts Permission to Open a Database Tunnel

6.3.3.4.91 list-ecm-repositories

Returns details and settings of all repositories in the specified subaccount.

Parameters

Required	
`-a, --account`	Subaccount technical name Specify an existing subaccount of which you are already a member. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Optional	
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Example

```
neo list-ecm-repositories --account acme --host hana.ondemand.com --user
<myemail@example.com>
```

```
ExampleRepository
  Display name      : Example Repository
  Description       : This is an example repository with Virus Scan enabled.
  ID                : cdb158efd4212fc00726b035
  Application       : Neo CLI
  Virus Scan        : on
```

```
ExampleRepositoryNoVS
  Display name      : Example Repository without Virus Scan
  Description       : This is an example repository with Virus Scan disabled.
  ID                : cdb158efd4212fc00726b035
  Application       : Neo CLI
  Virus Scan        : off
Number of Repositories: 2
```

6.3.3.4.92 list-jmx-checks

Lists JMX checks.

```
neo list-jmx-checks -a <subaccount_technical_name> -u <e-mail_or_user> -h <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.

Optional	
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) **i Note** If this parameter is used, only the application level checks are listed (the subaccount level checks are not listed).
`-R, --recursively`	Lists JMX checks recursively, starting from the specified level. For example, if only 'subaccount' is passed as an argument, it starts from the subaccount level and then lists all checks configured on application level. `Default`: false `Type`: boolean

i Note

If the optional parameters are not used, only the JMX checks on subaccount level are listed.

Example

Listing all the JMX checks on subaccount and application levels:

```
neo list-jmx-checks -a mysubaccount -u p1234567 -R -h hana.ondemand.com
```

Sample output:

```
Neo Console Client
Password for your user:
Running list-jmx-check with the following parameters:
    account    : mysubaccount
    host       : https://hana.ondemand.com
    recursively: true
    user       : p1234567
Subaccount-level JMX checks
    account         : mysubaccount
    check-name      : JVM Heap Memory Used
    object-name     : java.lang:type=Memory
    critical        : 60
    attribute       : HeapMemoryUsage
    attribute key   : used
    warning         : 700000000
    critical        : 900000000
    unit            : B
Application JMX checks

      application       : demo
      check-name        : JVM Heap Memory Used
      object-name       : java.lang:type=Memory
      attribute         : HeapMemoryUsage
      attribute key     : used
      warning           : 600000000
      critical          : 850000000
      unit              : B
```

Related Information

JMX Checks [page 738]
Monitoring Java Applications [page 729]

6.3.3.4.93 list-keystores

This command is used to list the available keystores. You can list keystores on subaccount, application, and subscription levels.

Parameters

To list all parameters available for this command, execute **`neo help list-keystores`** in the command line.

Required	
`-a, --account`	Consumer subaccount technical name The subaccount for which you provide username and password. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-b, --application`	Application name • Use `--application <consumer_application_name>` if the application is running in your subaccount. • Use `--application <provider_subaccount_technical_name>:<provider_application_name>` if the application is running in another subaccount. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Example

On Subscription Level

```
neo list-keystores --account <consumer_subaccount_technical_name> --application
<provider_subaccount_technical_name>:<provider_application_name>
--user <e-mail_or_user> --host hana.ondemand.com
```

On Application Level

```
neo list-keystores --account <consumer_subaccount_technical_name> --application
<consumer_application_name>
--user <e-mail_or_user> --host hana.ondemand.com
```

On Subaccount Level

```
neo list-keystores --account <consumer_subaccount_technical_name> --user <e-
mail_or_user> --host hana.ondemand.com
```

Related Information

Keystore Console Commands [page 1783]
Keys and Certificates [page 1781]
Using the Keystore Service for Client Side HTTPS Connections [page 1791]

6.3.3.4.94 list-loggers

This command lists all available loggers with their log levels for your application.

```
neo list-loggers  --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help list-loggers`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	Password for the specified user. To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Your email, SAP ID or user name `Type`: string

Example

```
neo list-loggers --account mysubaccount --application demo --user p1234567890 --
host hana.ondemand.com
```

Related Information

Using Logs in the Console Client
Exit Codes [page 1584]

6.3.3.4.95 list-logs

This command lists all log files of your application sorted by date in a table format, starting with the latest modified.

```
neo list-logs --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help list-logs`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	Password for the specified user. To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Your email, SAP ID or user name `Type`: string

Optional	
`-t`, `--apptype`	Application type. Use this parameter to differentiate applications with the same name from different technologies (Java and HTML5). The following values are allowed for the application type: • `java` - for Java applications • `html5` - for HTML5 applications `Default`: `java` `Type`: string

Example

```
neo list-logs --account mysubaccount --application demo --apptype html5 --user
p1234567890 --host hana.ondemand.com
```

Related Information

Using Logs in the Console Client
Exit Codes [page 1584]

6.3.3.4.96 list-mtas

This command lists the Multitarget Application (MTA) archives that are deployed to your subaccount or provided by another subaccount.

```
neo list-mtas --host <host> --account <subaccount_technical_name> --user <e-
mail_or_user>
```

Parameters

To list all parameters available for this command, execute **`neo help list-mtas`** in the command line.

Required	
`-a`, `--account`	The name of the subaccount for which you provide a user and a password.
`-h`, `--host`	The host on which you execute the command.

Required	
`-p, --password`	Your user password. We recommend that you enter it only when prompted, and not explicitly as a parameter in a properties file or the command line.
`-u, --user`	Your user e-mail or SAP ID (SCN) user name.

Optional	
Command-specific parameters	
`--available-for-subscription`	If you use this parameter, the command will list only the MTAs that are available for subscription to the corresponding subaccount. The MTAs, which are deployed by the subaccount, will not be listed.

Example

To check the deployed MTA archives to a given subaccount, execute:

```
neo list-mtas –host hana.ondemand.com --account mysubaccount --user
mymail@example.com
```

To check the MTAs that are available for subscription to a given subaccount, execute:

```
neo list-mtas –host hana.ondemand.com --account mysubaccount --user
mymail@example.com --available-for-subscription
```

6.3.3.4.97 list-mta-operations

This command shows the MTA operation status with a given ID.

```
neo list-mta-operations --host <host> --account <subaccount_technical_name> --
operation-id < operation-id > --user <e-mail_or_user>
```

Parameters

To list all parameters available for this command, execute **`neo help list-mta-operations`** in the command line.

Required	
`-a, --account`	The name of the subaccount for which you provide a user and a password.
`-h, --host`	The host on which you execute the command.

Required	
`-p, --password`	Your user password. We recommend that you enter it only when prompted, and not explicitly as a parameter in a properties file or the command line.
`-u, --user`	Your user e-mail or SAP ID (SCN) user name.
`--operation-id`	The ID of the operation. **i Note** This parameter is optional. If you do not use this parameter, all operations that have not been cleaned up within the last 24 hours will be listed.

Example

To check the deployment status of an MTA with ID "i2", execute:

```
neo list-mta-operations –host hana.ondemand.com --account mysubaccount --
operation-id i2 --user mymail@example.com
```

Related Information

deploy-mta [page 1429]

6.3.3.4.98 list-proxy-host-mappings

Lists the proxy hosts mapped to an application hostname.

```
neo list-proxy-host-mappings --account <subaccount_technical_name> --app-host
<application_host> --proxy <proxy_host:port> -h <host> -u <e-mail_or_user> -p
<password>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required	
`-h, --host`	Region host on which you execute the command. `Type`: URL. For acceptable values see Regions.
`-u, --user`	Your email, SAP ID, or username. `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Optional	
`--app-host`	Your application hostname. For example: `myapp.hana.ondemand.com`
`--proxy`	On-premise reverse proxy hostname and port. Separate proxy hostname and port with a colon (':'). For example: `loc.corp:123`

Example

```
neo list-proxy-host-mappings --account mysubaccount --app-host
app.hana.ondemand.com --proxy loc.corp:123 -h hana.ondemand.com -u username -p
******
```

An application with hostname `app.hana.ondemand.com` is mapped to proxy host `loc.corp:123`.

Related Information

Configuring Application Access via On-Premise Reverse Proxy [page 1668]
map-proxy-host [page 1513]
unmap-proxy-host [page 1575]

6.3.3.4.99 list-runtimes

The command displays all available application runtime containers.

```
neo list-runtimes --user <e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help list-runtimes`** in the command line.

Required	
`-u, --user`	Use your email, SAP ID, or username. `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.

Example

```
neo list-runtimes --user myuser --host hana.ondemand.com
```

Related Information

list-runtime-versions [page 1501]
Choose Application Runtime Version [page 1594]
View Runtime Information [page 1608]

6.3.3.4.100 list-runtime-versions

The command displays the supported application runtime container versions for your SAP BTP SDK for Neo environment. Only recommended versions are shown by default. You can also list supported version for a particular runtime container.

```
neo list-runtime-versions --user <e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help list-runtime-versions`** in the command line.

Required	
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.

Optional	
`--all`	Lists all supported application runtime container versions. Using a previously released runtime version is not recommended.
`--runtime`	Lists supported version only for the specified runtime container. For more information, see Application Runtime Container [page 857].

Example

```
neo list-runtime-versions --user myuser --host hana.ondemand.com --runtime neo-
java-web
```

Related Information

Choose Application Runtime Version [page 1594]
View Runtime Information [page 1608]
list-runtimes [page 1500]

6.3.3.4.101 list-schemas

This command lists all schemas contained in the specified subaccount.

```
neo list-schemas -a <subaccount_technical_name> -h <host> -u <e-mail_or_user>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Optional	
`--verbose`	Displays additional information about each schema: database type and database version `Default`: *off* `Type`: switch, takes no value

Example

```
neo list-schemas -a mysubaccount -h hanatrial.ondemand.com -u mymail@example.com
--verbose
```

Related Information

Example Scenarios
Administering Database Schemas

6.3.3.4.102 list-schema-access-grants

This command lists all current schema access grants for a specified subaccount.

```
neo list-schema-access-grants --host <host> --account
<subaccount_technical_name> --user <e-mail_or_user>
```

Note that the list does not include grants that have been revoked.

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Optional	
`-i, --id`	Schema ID Lists only the grants for the specified schema `Type`: string

Example

```
neo list-schema-access-grants --host hanatrial.ondemand.com --account
mysubaccount --user mymail@example.com --id myschema
```

Related Information

Grant Access to Schemas
grant-schema-access [page 1452]
revoke-schema-access [page 1533]

6.3.3.4.103 list-security-rules

This console client command lists the security group rules configured for a virtual machine.

```
neo list-security-rules --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --name <vm_name>
```

Parameters

Required	
`-a, --account`	Your subaccount technical name. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-n, --name`	Type the name of the virtual machine. `Type`: string `Condition`: Use either `--id` or `--name`
`-i , --id`	ID of the virtual machine `Type`: string `Condition`: Use either `--id` or `--name`
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Example

```
neo list-security-rules --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --name myvm
```

As an output of the `list-security-rules` command, you may receive the HANA or JAVA source types previously created with the `create-security-rule` command, or an internally managed security group rule of type CIDR for a registered access point. The security group rule of type CIDR allows communication between the load balancer of the platform and the virtual machine.

Related Information

Manage Network Communication for SAP Virtual Machine Service
create-security-rule [page 1394]

6.3.3.4.104 list-ssh-tunnels

Lists the currently opened SSH tunnels on the user's machine.

```
neo list-ssh-tunnels
```

Related Information

open-ssh-tunnel [page 1516]
close-ssh-tunnel [page 1376]

6.3.3.4.105 list-ssl-hosts

Lists SSL hosts for a given subaccount.

```
neo list-ssl-hosts --account <subaccount_technical_name> --user <e-mail_or_user>
--host <host>
```

If you have several subaccounts in your global account, the `list-ssl-hosts` command returns all SSL hosts in these subaccounts.

Parameters

To list all parameters available for this command, execute **`neo help list-ssl-hosts`** in the command line.

Required	
`-a`, `--account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required	
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

```
neo list-ssl-hosts --account mysubaccount --user mymail@example.com --host
hana.ondemand.com
```

Related Information

create-ssl-host [page 1395]
Create an SSL Host [page 1651]

6.3.3.4.106 list-subscribed-accounts

Lists all subaccounts subscribed to a given Java application.

```
neo list-subscribed-accounts --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help list-subscribed-accounts`** in the command line.

Required	
`-a, --account`	Subaccount technical name This is the subaccount of the application provider. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name To be able to execute this command, the specified user must be a member of the provider subaccount. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Example

```
neo list-subscribed-accounts --account mysubaccount --application demo --user
myuser --host us1.hana.ondemand.com
```

Related Information

Subscribe to Java Multitenant Applications in the Neo Environment [page 838]
Subscribe to HTML5 Multitenant Applications in the Neo Environment [page 840]
subscribe [page 1566]
unsubscribe [page 1577]
list-subscribed-applications [page 1509]

6.3.3.4.107 list-subscribed-applications

Lists all Java applications to which a given subaccount is subscribed.

```
neo list-subscribed-applications --account <subaccount_technical_name> --user <e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help list-subscribed applications`** in the command line.

Required	
`-a, --account`	Subaccount technical name This is the subaccount of the applications consumer. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name To be able to execute this command, the specified user must be a member of the subaccount. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Example

```
neo list-subscribed-applications --account consumersubaccount --user myuser --host hana.ondemand.com
```

Related Information

Subscribe to Java Multitenant Applications in the Neo Environment [page 838]
Subscribe to HTML5 Multitenant Applications in the Neo Environment [page 840]

subscribe [page 1566]
unsubscribe [page 1577]
list-subscribed-accounts [page 1507]

6.3.3.4.108 list-vms

Lists all virtual machines in the specified subaccount. You can get information for a concrete virtual machine by name. The command output lists information about the virtual machine, such as size; status; SSH key; floating IP (if assigned); volume IDs.

```
neo list-vms --account <subaccount_technical_name> --user <e-mail_or_user> --
host <host> --name <vm_name>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Optional	
`-n, --name`	Name of the virtual machine `Type`: string

Example

```
neo list-vms --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --name myvm
```

Related Information

Manage Virtual Machines

6.3.3.4.109 list-volumes

Lists all volumes in the specified subaccount.

```
neo list-volumes --account <subaccount_technical_name> --user <e-mail_or_user> --
host <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Optional	
`-n, --not-attached-only`	Lists only volumes that are not attached.

Example

```
neo list-volumes --account mysubaccount --host hana.ondemand.com --user
myemail@example.com
```

Related Information

Manage Volume Snapshots

6.3.3.4.110 list-volume-snapshots

Lists all volume snapshots in the specified subaccount. Use `display-volume-snapshot` to get information about a specific volume snapshot.

```
neo list-volume-snapshots --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Optional	
`-v, --volume-id`	Unique identifier of a volume. If specified, only volume snapshots created from this volume will be displayed. `Type`: string

Example

```
neo list-volume-snapshots --account mysubaccount --host hana.ondemand.com --user
myemail@example.com
```

Related Information

Manage Volume Snapshots
create-volume-snapshot [page 1399]
display-volume-snapshot [page 1438]
delete-volume-snapshot [page 1421]

6.3.3.4.111 map-proxy-host

Maps an application host to an on-premise reverse proxy host and port.

```
neo map-proxy-host --account <subaccount_technical_name> --app-host
<application_host> --proxy <proxy_host:port> -h <host> -u <e-mail_or_user> -p
<password>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`--app-host`	Your application hostname. For example: `myapp.hana.ondemand.com`
`--proxy`	On-premise reverse proxy hostname and port. Separate proxy hostname and port with a colon (':'). For example: `loc.corp:123`
`-h, --host`	Region host on which you execute the command. `Type`: URL. For acceptable values see Regions.
`-u, --user`	Your email, SAP ID, or username. `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Example

```
neo map-proxy-host --account mysubaccount --app-host app.hana.ondemand.com --
proxy loc.corp:123 -h hana.ondemand.com -u username -p ******
```

An application with hostname `app.hana.ondemand.com` is mapped to proxy host `loc.corp:123`.

Related Information

Configuring Application Access via On-Premise Reverse Proxy [page 1668]
unmap-proxy-host [page 1575]
list-proxy-host-mappings [page 1499]

6.3.3.4.112 open-db-tunnel

This command opens a database tunnel to the database system associated with the specified schema or database.

> **i Note**
>
> Make sure that you have installed the required tools correctly.
>
> If you face trouble using this command, please check that your installation is correct.
>
> For more information, see Set Up the Console Client [page 847] and Using the Console Client [page 1348].

The command has two modes:

- Default mode: The tunnel remains open until you explicitly close it by pressing `ENTER` in the command line. It is closed automatically after 24 hours or if the command window is closed.

  ```
  neo open-db-tunnel -a <subaccount_technical_name> -h <host> -u <user> -i
  <ID_or_alias>
  ```

- Background mode: The database tunnel is opened in a separate process. Use the `close-db-tunnel` command to close the tunnel once you are done, or it is closed automatically after one hour.

  ```
  neo open-db-tunnel -a <subaccount_technical_name> -h <host> -u <user> -i
  <ID_or_alias> --background
  ```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-i, --id`	• SAP ASE database system (`ASE`): Specify the database ID of an SAP ASE user database. • SAP HANA tenant database system (`HANAMDC`): Specify the database ID of an SAP HANA tenant database. • SAP HANA single-container database system (`HANAXS`): Specify the alias of the database system. • Shared SAP HANA database: Specify the schema ID of a schema. `Type`: string
`--access-token`	Identifies a database access permission. The access token and database ID parameters are mutually exclusive.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Optional	
`--background`	Opens the tunnel session in a background process `Type`: switch, takes no value
`--output`	Prints information about the tunnel in a special output format. `Acceptable values`: 'json' `Type`: string

Example

```
neo open-db-tunnel -a mysubaccount -h hanatrial.ondemand.com -u
mymail@example.com -i mydatabase
```

Related Information

Accessing Databases Remotely
Open Database Tunnels
Connect to SAP HANA Databases via the Eclipse IDE
Connect to SAP HANA Schemas via the Eclipse IDE (Neon)
close-db-tunnel [page 1376]
Connect to the Remote SAP ASE Database
Automating the Use of Database Tunnels
Machine-Readable Command Output [page 1353]
Connect DB Tools to SAP HANA via Service Channels [page 447]

6.3.3.4.113 open-ssh-tunnel

Opens a secure tunnel to a specific virtual machine.

```
neo open-ssh-tunnel --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --vm-id <vm_id>
```

or

```
neo open-ssh-tunnel --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --vm-name <vm_name>
```

> **i Note**
>
> The tunnel is closed automatically after 24 hours or if the command window is closed.

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-n, --vm-name`	Name of the virtual machine `Type`: string `Condition`: Use either `--vm-id` or `--vm-name`

Required	
`-i ,--vm-id`	ID of the virtual machine `Type`: string `Condition`: Use either `--vm-id` or `--vm-name`
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
Optional	
`-r, --port`	Port on which you want to open the SSH tunnel If not specified, the command opens the tunnel on a random port.

Example

```
neo open-ssh-tunnel --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --vm-id myvmid
```

or

```
neo open-ssh-tunnel --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --vm-name myvmname
```

Related Information

list-ssh-tunnels [page 1506]
close-ssh-tunnel [page 1376]

6.3.3.4.114 put-destination

This command uploads destination configuration properties files and JKS files. You can upload them on subaccount, application or subscribed application level.

```
neo put-destination --account <subaccount_technical_name> --user <e-
mail_or_user> --localpath <destination_file_or_JKS_file_localpath> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help put-destination`** in the command line.

Required	
`-a`, `--account`	Your subaccount. The subaccount for which you provide username and password. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b`, `--application`	The application for which you upload a destination. Cases: • Use `--application <myapp>` if the application is running in your subaccount. • Use `--application <provider_subaccount>:<provider_app>` if the application is running in another subaccount. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h`, `--host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`--localpath`	The path to a destination or a JKS file on your local file system. `Type`: string
`-p`, `--password`	Password for the specified user. To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string **i Note** When uploading a destination configuration file that contains a password field, the password value remains available in the file. However, if you later download this file, using the `get-destination` command, the password value will no more be visible. Instead, after `Password =...`, you will only see an empty space.
`-u`, `--user`	Your email, SAP ID or user name `Type`: string

Examples

- To upload a destination on subaccount level, execute:

```
neo put-destination --account mysubaccount --user p1234567890 --localpath C:
\myfiles\myconfiguration.jks --host hanatrial.ondemand.com
```

- To upload a destination on application level, execute:

```
neo put-destination --account mysubaccount --user p1234567890 --application
demo --localpath C:\SDK\tools\samples\connectivity\weather --host
hanatrial.ondemand.com
```

- To upload a destination on subscribed application level, execute:

```
put-destination -h <host> -a <subaccount> -u <user> -b
<provider_subaccount>:<application> --localpath <path to destination file>
```

Related Information

Upload Destinations [page 58]
Exit Codes [page 1584]

6.3.3.4.115 reboot-vm

Reboots a virtual machine by name or by ID. By default, the reboot is soft. You can perform a hard reboot if you use the `--hard` parameter.

```
neo reboot-vm --account <subaccount_technical_name> --user <e-mail_or_user> --
host <host> --name <vm_name>
```

or

```
neo reboot-vm --account <subaccount_technical_name> --user <e-mail_or_user> --
host <host> --id <vm_id>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-n, --name`	Name of the virtual machine `Type`: string `Condition`: Use either `--id` or `--name`

Required	
`-i, --id`	ID of the virtual machine `Type`: string `Condition`: Use either `--id` or `--name`
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Optional	
`--hard`	Performs a hard reboot of the specified virtual machine. The hard reboot lets you force a shutdown before restarting the virtual machine. `Type`: switch; takes no value `Default`: soft. The soft reboot attempts to shut down gracefully and restart the virtual machine.

Examples

- If you want to perform a soft reboot, execute one of the following two commands:

```
neo reboot-vm --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --name myvm
```

```
neo reboot-vm --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --id myvmid
```

- If you want to perform a hard reboot, execute one of the following two commands:

```
neo reboot-vm --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --name myvm --hard
```

```
neo reboot-vm --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --id myvmid --hard
```

Related Information

Manage Virtual Machines
list-vms [page 1510]

6.3.3.4.116 register-access-point

Registers an access point URL for a virtual machine specified by name or ID.

```
neo register-access-point --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --name <vm_name>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-i , --id`	ID of the virtual machine `Type`: string `Condition`: Use either `--id` or `--name`
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-n, --name`	Name of the virtual machine `Type`: string `Condition`: Use either `--id` or `--name`
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Example

```
neo register-access-point --account mysubaccount --user mymail@example.com --
host hana.ondemand.com --name myvm
```

The `register-access-point` command creates an internally managed security rule of type CIDR, which allows communication between the load balancer of the platform and the virtual machine.

Related Information

Enable Internet Access
unregister-access-point [page 1576]

6.3.3.4.117 remove-ca

Removes trusted CAs from a bundle or deletes a whole bundle and all certificates in it.

```
neo remove-ca --account <subaccount_technical_name> --host <host> --user <e-
mail_or_user> --bundle <bundle_name>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or username. `Type`: string
`--bundle`	Name of a new or existing bundle in which CAs will be added. A bundle can hold up to 120 certificates. `Type`: string The name of a bundle must start with a letter and can only contain 'a' - 'z' 'A' - 'Z' '0' - '9' ".", "_" and "-".

Optional	
`--serial-number`	A comma-separated (,) list of serial numbers of the CA certificates to be removed.

Optional	
`--expired`	Removes all expired trusted CA certificates in the specified bundle. Takes no value.

Example

```
neo remove-ca --account mysubaccount --host hana.ondemand.com --user
mymail@example.com --bundle mybundle
```

Related Information

add-ca [page 1360]
list-cas [page 1483]
set-ssl-host [page 1552]
Managing Client Certificate Authentication for Custom Domains [page 1660]

6.3.3.4.118 remove-custom-domain

Removes a custom domain as an access point of an application. Use this command if you no longer want an application to be accessible on the configured custom domain.

```
neo remove-custom-domain --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --custom-domain <custom_domain> --ssl-host <ssl_host>
```

Parameters

To list all parameters available for this command, execute **`neo help remove-custom-domain`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.

Required	
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-e, --custom-domain`	Custom domain for accessing the application `Type`: string (Fully qualified domain name - FQDN)
`-l, --ssl-host`	SSL host as defined with the `--name` parameter when created, or 'default' if not specified.

Example

```
neo remove-custom-domain --account mysubaccount --user mymail@example.com --host
hana.ondemand.com --custom-domain www.example.com --ssl-host mysslhostname
```

Related Information

add-custom-domain [page 1361]
Add the Custom Domain [page 1655]
Configuring Custom Domains [page 1649]

6.3.3.4.119 remove-platform-domain

Removes a platform domain (under hana.ondemand.com) as an access point for an application.

```
neo remove-platform-domain --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user> --host <host> --platform-domain
<platform_domain>
```

Parameters

To list all parameters available for this command, execute **`neo help remove-platform-domain`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-m, platform-domain`	Platform domain under hana.ondemand.com `Type`: URL

Example

```
neo remove-platform-domain --account mysubaccount --application myapp --user
myuser --host hana.ondemand.com --platform-domain svc.hana.ondemand.com
```

Related Information

add-platform-domain [page 1363]
Using Platform Domains [page 1666]

6.3.3.4.120 reset-ecm-key

If you have forgotten the repository key, use this command to request a new repository key.

This command only creates a new key that replaces the old one. You cannot use the old key any longer. The command does not affect any other repository setting, for example, the virus scan definition. If you just want to change your current repository key, use the `edit-ecm-repository` command.

```
neo reset-ecm-key --name <repository_name> --account <subaccount_technical_name>
--host <host> --user <e-mail_or_user>
```

Parameters

Required	
`-a, --account`	Subaccount technical name Specify an existing subaccount of which you are already a member. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-n, --name`	Name of the repository `Type`: string

Example

```
neo reset-ecm-key --name com.foo.MyRepository --account sap --host
hana.ondemand.com --user p1940248318
```

This example resets the repository key for the `com.foo.MyRepository` repository and creates a new repository key, for example `fp0TebRs14rwyqq`.

Related Information

edit-ecm-repository [page 1441]

6.3.3.4.121 reset-log-levels

This command resets all logger levels to their initial state.

```
neo reset-log-levels  --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help reset-log-levels`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	Password for the specified user. To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Your email, SAP ID or user name `Type`: string

Example

```
neo reset-log-levels --account mysubaccount --application demo --user
p1234567890 --host hana.ondemand.com
```

Related Information

Using Logs in the Console Client
Exit Codes [page 1584]

6.3.3.4.122 restart

Use this command to restart your application or a single application process. The effect of the restart command is the same as executing the stop command first and when the application is stopped, starting it with the start command.

```
neo restart --account <subaccount_technical_name> --application
<application_name> --host <host> --user <e-mail_or_user>
```

```
neo restart --application-process-id <ID> --user <e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute the **`neo help restart`** command.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) `Condition`: Not required if using `--application-process-id`
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) `Condition`: Not required if using `--application-process-id`
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-y, --synchronous`	Triggers the process and waits until the application is restarted. The command without the `--synchronous` parameter triggers the restarting process and exits immediately without waiting for the application to start. `Default`: off `Type`: switch, takes no value

Optional	
`-i, --application-process-id`	Unique ID of a single application process. Use it to restart a particular application process instead of the whole application. As the process ID is unique, you do not need to specify subaccount and application parameters. You can list the application process ID by using the `<status>` command. `Default`: none `Type`: string (hexadecimal sequence of 2 to 40 characters)

Example

To restart the whole application and wait for the operation to finish, execute:

```
 neo restart --account mysubaccount --application myapp --user
mymail@example.com --host hana.ondemand.com --synchronous
```

Related Information

stop [page 1562]
status [page 1555]
Exit Codes [page 1584]

6.3.3.4.123 restart-hana

Restarts an SAP HANA system or an SAP HANA database service.

> **i Note**
>
> To use this command, log on with a user with administrative rights for the subaccount.

> **i Note**
>
> The `restart-hana` operation will be executed asynchronously. Temporary downtime is expected for SAP HANA database system or SAP HANA XS Engine, including inability to work with SAP HANA studio, SAP HANA Web-based Development Workbench and Cockpit UIs dependent on SAP HANA XS.

This command has two alternative uses:

- For restarting the entire SAP HANA database system (including all tenant databases when working with SAP HANA tenant database systems):

  ```
  neo restart-hana --host <host> --account <subaccount_technical_name> --user
  <e-mail_or_user> --id <SAP HANA system identifier> --system
  ```

- For restarting an SAP HANA database service (`indexserver`, `xsengine`)

```
neo restart-hana --host <host> --account <subaccount_technical_name> --user
<e-mail_or_user> --id <SAP HANA system identifier> --service-name
<service_name>
```

After you trigger the command, you can monitor the command execution in SAP HANA Studio, using *Configuration and Monitoring* > *Open Administration*.

Parameters

To list all parameters available for this command, execute **`neo help restart-hana`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-i, --id`	The ID of a productive SAP HANA database system **i Note** You can find the SAP HANA database system ID using the list-dbms [page 1487] command or in the *Databases & Schemas* section in the cockpit by navigating to *SAP HANA / SAP ASE* > *Databases & Schemas*. It must start with a letter and can contain uppercase and lowercase letters ('a' - 'z', 'A' - 'Z'), numbers ('0' - '9'), and the special characters '.' and '-'. `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Your e-mail, SAP ID, or SCN user name `Type`: string
`--service-name`	The SAP HANA database service to be restarted. You can choose between the following values: • *xsengine* - for restarting the SAP HANA XS service • *indexserver* - for restarting the SAP HANA index server
`--system`	If available, the entire SAP HANA database system will be restarted.

Example

To restart the SAP HANA database system with ID `myhanaid` running on the productive host, execute:

```
neo restart-hana --system --id myhanaid --account mysubaccount --host
hana.ondemand.com --user mymail@example.com
```

To restart the SAP XS Engine service on SAP HANA database system with ID `myhanaid`, execute:

```
neo restart-hana --service-name xsengine --id myhanaid --account mysubaccount --
host hana.ondemand.com --user mymail@example.com
```

Related Information

Restart Database Systems
SAP HANA Administration Guide
SAP BTP Release Notes

6.3.3.4.124 revoke-db-access

This command revokes the database access permissions given to another subaccount.

```
neo revoke-db-access --account <subaccount_ID> --user <e-mail_or_user> --host
<host> --id <database_ID>
--to-account <to-subaccount_technical_name> --permissions <permission_type>
```

Parameters

Required	
`-a, --account`	ID of the subaccount providing the database. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Required	
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.

Optional	
`-i, --id`	ID of the database to which access will be revoked.
`-to-account`	The subaccount from which access permissions will be revoked.
`-permissions`	Comma-separated list of ther permissions to be revoked. Acceptable values: 'TUNNEL', 'BINDING'.

Example

```
neo revoke-db-access --account myProviderSubaccount --user mymail@example.com --
host hana.ondemand.com --id myDB --to-account myConsumerSubaccount --permissions
BINDING
```

Related Information

grant-db-access [page 1450]
list-db-access-permissions [page 1485]
Managing Access Permissions
Sharing Databases in the Same Global Account

6.3.3.4.125 revoke-db-tunnel-access

This command revokes database access that has been given to another subaccount.

```
neo revoke-db-tunnel-access -h <host> -u <user> -a <my subaccount> --access-
token <token>
```

Parameters

Required	
`-- access-token`	Access token that identifies the permission to access the database `Type`: string
`--silent`	(optional) Suppresses the command line confirmation prompt `Type`: boolean

Optional	
`--output`	Confirmation that the permission for opening the database tunnel from the other subaccount to the database was successfully revoked `Type`: string

Example

```
neo revoke-db-tunnel-access -h hanatrial.ondemand.com -u mymail@example.com -a
mysubaccount --access-token 31t0dpim6rtxa00wx5483vqe7in8i3c1phv759w9oqrutf638l
```

Related Information

grant-db-tunnel-access [page 1452]
list-db-tunnel-access-grants [page 1490]
Revoke Tunnel Access to Databases for Other Subaccounts

6.3.3.4.126 revoke-schema-access

This command revokes the schema access granted to an application in another account.

```
neo revoke-schema-access --host <SAP HANA Cloud host> --account <subaccount
technical name> --user <e-mail or user name> --access-token <access token>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`--access-token`	Access token that identifies the grant. Grants can only be revoked by the granting subaccount.

Example

```
neo revoke-schema-access --host hanatrial.ondemand.com --account mysubaccount --
user mymail@example.com --access-token
vm6431dhjcr2e3dbt0fk6jpzm2w7oo3q48yumf1c6uu8b9pt9z
```

Related Information

grant-schema-access [page 1452]
list-schema-access-grants [page 1503]
Managing Access to Databases for Other Subaccounts
Sharing Databases with Other Subaccounts
Grant Access to Schemas

6.3.3.4.127 rolling-update

The rolling-update command performs update of an application without downtime in one go.

Prerequisites

- You have at least one application process that is not in use, see your compute unit quota.
- The command can be used with compatible application changes only.

The rolling-update command performs the following steps:

1. Deploys a new version of the application.

   > **i** Note
   >
   > If you use enhanced disaster recovery, the application is also deployed on the disaster recovery region without being started.

2. Starts a new application process.
3. Disables new connection requests for one of the old application processes.
4. Waits for the given timeout.
5. Stops the disabled application process.
6. Repeats steps 2 to 5 for all remaining old application processes.

```
neo rolling-update --host <host> --account <subaccount_technical_name> --
application <application_name>
--source <file_location> --user <e-mail_or_user>
```

Parameters

To list all parameters available for this command, execute **`neo help rolling-update`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-s, --source`	A comma-separated list of file locations, pointing to WAR files, or folders containing them If you want to deploy more than one application on one and the same application process, put all WAR files in the same folder and execute the deployment with this source, or specify them as a comma-separated list. `Type`: file location
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions.

Required

`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional

`--compression`	Enable or disable gzip response compression `Default`: *off* Possible values: *on* (allow compression), *off* (disable compression), *force* (forces compression for all responses) or an *integer* (which enables compression and specifies the compression-min-size value in bytes). For more information, see Enable and Configure Gzip Response Compression [page 1597]
`--compressible-mime-type`	A comma separated list of MIME types for which compression will be used `Default`: *text/html, text/xml, text/plain* `Condition:` applicable if compression is enabled
`--compression-min-size`	Responses bigger than this value get compressed `Condition:` applicable if compression is enabled
`--connections`	The number of connections used to deploy an application. Use it to speed up deployment of application archives bigger than 5 MB in slow networks. Choose the optimal number of connections depending on the overall network speed to the cloud. `Default`: *2* `Acceptable values`: **`1-6`** `Type`: integer
`--ev`	Environment variables for configuring the environment in which the application runs. Sets one environment variable by removing the previously set value; can be used multiple times in one execution. `Type`: --ev <KEY1>=<VALUE1> --ev <KEY2>=<VALUE2> , where a key-value pair specifies one environment variable If you provide a key without any value (--ev <KEY1>=), the `-ev` parameter is ignored. For a value that contains spaces, use quotation marks.

Optional

`-j, --java-version`	JRE version `Default`: depends on the SDK for SAP BTP, Neo environment `Type`: the version number of the JRE 7. For more information, see Choose JRE Version [page 1595]
`--timeout`	Timeout before stopping the old application processes (in seconds) `Default`: 60 seconds
`-V, --vm-arguments`	System properties (-D<name>=<value>) separated with space that will be used when starting the application process. Memory settings of your compute units. You can set the following memory parameters: -Xms, -Xmx, -XX:PermSize, -XX:MaxPermSize. We recommend that you use the default memory settings. Change them only if necessary and note that this may impact the application performance or its ability to start. For more information, see Configure VM Arguments [page 1598]
`-z, --size`	Compute Unit size: lite, pro, prem, prem-plus The compute unit size defines the default memory settings. For more information, see Compute Units [page 867] `Default`: *lite*
`--runtime-version`	The runtime version on which the application will be started and will run on the same version after a restart. Otherwise, by default, the application is started on the latest minor version (of the same major version) which is backward compatible and includes the latest corrections (including security patches), enhancements, and updates. Note that choosing this option does not affect already started application processes. You can view the recommended versions by executing the *list-runtime-versions* command. **i Note** If you choose your runtime version, consider its expiration date and plan updating to a new version regularly. For more information, see Choose Application Runtime Version [page 1594]
`--uri-encoding`	Specifies the character encoding used to decode the URI bytes on application request. `Default`: *ISO-8859-1* For more information, see the encoding sets supported by Java SE 6 and Java SE 7 .

Example

```
neo rolling-update --host us1.hana.ondemand.com --account mysubaccount --
application myapp --source samples/deploy_war/example.war
--user mymail@example.com --timeout 5
```

Related Information

deploy [page 1422]
Update Applications with Zero Downtime [page 1610]

6.3.3.4.128 sdk-upgrade

Use this command to upgrade the SAP BTP SDK for Neo environment that you are currently working with.

```
neo sdk-upgrade
```

The command checks for a more recent version of the SDK and then upgrades the SDK. There are two possible cases:

- Your SDK version is up to date and no upgrade is needed.
- Your SDK version is not up to date and an upgrade process is triggered.
  Then an upgrade of the SDK is triggered. The old SDK is backed up in case something goes wrong with the upgrade.

  > **i Note**
  >
  > All files and servers that you add to your SDK will be preserved during upgrade.

Example

```
neo sdk-upgrade
```

6.3.3.4.129 set-alert-recipients

Sets alert recipients.

Prerequisites

- You have set up the console client.
  For more information, see Set Up the Console Client [page 847].
- The *manageMonitoringConfiguration* scope is assigned to the used platform role for the subaccount. For more information, see Platform Scopes [page 1307].

Overview

- Setting alert recipients becomes effective after five minutes.
- Setting additional recipients works only with the overwrite parameter (`-w` or `--overwrite`). In such a case, you have to overwrite the existing list of recipients with a new one.
- Setting alert recipients for a Java, HTML5, or SAP HANA XS application triggers sending all alerts for this application.
  Moreover, when the alert is for a Java application, the alert email for JMX checks and default metrics also includes the process ID. The alerts for availability checks and other application types don't include the process ID.
- Setting alert recipients on subaccount level triggers sending all alerts for all applications and database systems in this subaccount. In this case, alerts are sent for any type of a database system.

Email Confirmation Requirement

To comply with security requirements, we ask that email recipients confirm their email address before they can receive alert notifications. If users don't confirm their email address within 2 days, their email address will be removed from the list of alert recipients and they will not receive alert notifications. Users receive emails for confirmation when you set their email addresses as alert recipients. If you set additional recipients with the overwrite parameter, only the new recipients will receive a confirmation email. However, clearing alert recipients and then setting them again triggers emails for confirmation to the recipients again.

```
neo set-alert-recipients -a <subaccount_technical_name> -u <e-mail_or_user> -e
<recipient_emails> -h <host>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) When you configure alerting on subaccount level by omitting the `-b` or `--application` parameter, alerts are sent for all applications and database systems in the subaccount.
`-u, --user`	Use your email, SAP ID, or user name `Type`: string
`-e, --email`	Comma-separated list of recipient e-mails We recommend that you use distribution lists rather than personal email addresses. Keep in mind that you're responsible for handling of personal email addresses with respect to data privacy regulations applicable. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.

Optional	
`-b, --application`	Application name for Java or HTML5 applications, or productive SAP HANA instance database name and application name in the format `<instance name>:<application name>` for SAP HANA XS applications `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-w, --overwrite`	Overwrites existing recipients `Default`: false `Type`: boolean

> ⚠ Caution
>
> If you stop your application, you won't receive a notification alert for it because the alerting is suppressed with the manual stop of an application. Alerting is automatically enabled once again when you start the application.

Example

```
neo set-alert-recipients -a mysubaccount -b demo -u p1234567 -e
alert_recipients@example.com --host hana.ondemand.com
```

Related Information

Monitoring Java Applications [page 729]
Monitoring Database Systems [page 760]
list-alert-recipients [page 1479]
clear-alert-recipients [page 1373]

6.3.3.4.130 set-application-property

Use this command to change the value of a single property of a deployed application without the need to redeploy it. Execute the command separately for each property that you want to set. For the changes to take effect, restart the application.

To execute the command successfully, you need to to specify the new value of one property from the optional parameters table below.

```
neo set-application-property --host <host> --account <subaccount_technical_name>
--application <application_name> --user <e-mail_or_user>
--<property> <new_property_value>
```

Parameters

To list all parameters available for this command, execute the **`neo help set-application-property`** in the command line.

Required	
`-a`, `--account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b`, `--application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required

`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional

Command-specific parameters	
`--ev`	Environment variables for configuring the environment in which the application runs. Sets the new environment variable without removing the previously set value; can be used multiple times in one execution. `Type`: --ev <KEY1>=<VALUE1> --ev <KEY2>=<VALUE2>, where a key-value pair specifies one environment variable. If you provide a key without any value (--ev <KEY1>=), the environment variable KEY1 will be deleted. For a value that contains spaces, use quotation marks.
`-j, --java-version`	Java Virtual Machine version `Default`: depends on the SAP BTP SDK for Neo environment `Type`: the version number of the JRE 7. (beta) You can use JRE 8 with the Java Web Tomcat 7 runtime (neo-java-web version 2.25 or higher) in subaccounts enabled for beta features. For more information, see Choose JRE Version [page 1595]
`-m, --minimum-processes`	Minimum number of application processes, on which the application can be started `Default`: *1*
`-M, --maximum-processes`	Maximum number of application processes, on which the application can be started `Default`: *1*

Optional	
`-V, --vm-arguments`	Java Virtual Machine arguments System properties (-D<name>=<value>) separated with space that will be used when starting the application process. Memory settings of your compute units. You can set the following memory parameters: -Xms, -Xmx, -XX:PermSize, -XX:MaxPermSize. We recommend that you use the default memory settings. Change them only if necessary and note that this may impact the application performance or its ability to start. For more information, see Configure VM Arguments [page 1598]
`-z, --size`	Compute unit size `Acceptable values`: lite, pro, prem, prem-plus `Default`: the smallest size from the subaccount quotas For more information, see Compute Units [page 867]
`--runtime-version`	SAP BTP runtime version on which the application will be started and will run on the same version after a restart. Otherwise, by default, the application is started on the latest minor version (of the same major version) which is backward compatible and includes the latest corrections (including security patches), enhancements, and updates. Note that choosing this option does not affect already started application processes. You can view the recommended versions by executing the *list-runtime-versions* command. **i Note** If you choose your runtime version, consider its expiration date and plan updating to a new version regularly. For more information, see Choose Application Runtime Version [page 1594]
Tomcat connector attributes	
`--compression`	Enable or disable gzip response compression `Default`: *off* Possible values: *on* (allow compression), *off* (disable compression), *force* (forces compression for all responses) or an *integer* (which enables compression and specifies the compression-min-size value in bytes). For more information, see Enable and Configure Gzip Response Compression [page 1597]
`--compressible-mime-type`	A comma separated list of MIME types for which compression will be used `Default`: *text/html, text/xml, text/plain* `Condition:` applicable if compression is enabled
`--compression-min-size`	Responses bigger than this value get compressed `Condition:` applicable if compression is enabled

Optional	
`--connection-timeout`	Defines the number of milliseconds to wait for the request URI line to be presented after accepting a connection. `Default:` *20000*
`--max-threads`	Specifies the maximum number of simultaneous requests that can be handled. `Default:` *200*
`--uri-encoding`	Specifies the character encoding used to decode the URI bytes on application request. `Default:` *ISO-8859-1* For more information, see the encoding sets supported by Java SE 6 and Java SE 7 .

Example

To change the minimum number of server processes on which you want your deployed application to run, execute:

```
neo set-application-property --host hana.ondemand.com --account mysubaccount --
application myapp --user mymail@example.com --minimum-processes 2
```

Related Information

Update Application Properties [page 1593]
deploy [page 1422]
display-application-properties [page 1432]
restart [page 1528]
Org Administration Using the Cockpit

6.3.3.4.131 set-db-properties-ase

This command changes the properties for an ASE database.

```
neo set-db-properties-ase -a <subaccount_technical_name> -h <host> -u <e-
mail_or_user> -i <database_ID> --db-size <database_size>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-i, --id`	ASE database ID `Type`: string
`--db-size`	Size of the database in MB Specify a size that is greater than the actual size. **i Note** This parameter sets the maximum database size. The minimum database size is 24 MB. You receive an error if you enter a database size that exceeds the quota for this database system. The size of the transaction log will be at least 25% of the database size you specify.

Example

```
neo set-db-properties-ase -a mysubaccount -h hana.ondemand.com -u
mymail@example.com -i mydb --db-size dbsize
```

6.3.3.4.132 set-db-properties-hana

This command changes the properties for a SAP HANA database enabled for multitenant database container support.

```
neo set-db-properties-hana -a <subaccount_technical_name> -h <host> -u <e-
mail_or_user> -i <database_ID>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-i, --id`	HANA database ID `Type`: string

Optional	
`--web-access`	Enables or disables access to the HANA database from the Internet: 'enabled' (default), 'disabled'

Example

```
neo set-db-properties-hana -a mysubaccount -h hana.ondemand.com -u
mymail@example.com -i mydb
```

6.3.3.4.133 set-db-user-password-ase

This command sets a new password for an existing ASE database user and overwrites the existing password.

```
neo set-db-user-password-ase -a <subaccount_technical_name> -h <host> -u <e-
mail_or_user> -i <database_ID> --db-user <dbuser> --db-password
<database_user_password>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-i, --id`	ASE database ID `Type`: string
`--db-user`	Name of the ASE database user `Type`: string (up to 30 characters, starting with a letter)
`--db-password`	New password of the ASE database user (optional, queried at the command prompt if omitted). To protect your database password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Example

```
neo set-db-user-password-ase -a mysubaccount -h hana.ondemand.com -u
mymail@example.com -i mydb --db-user mydbuser --db-password SECRET
```

6.3.3.4.134 set-downtime-app

This command configures a custom downtime page (downtime application) for an application. The downtime page is shown to the user in the event of unplanned downtime of the original application.

```
neo set-downtime-app --account <subaccount_technical_name> --application
<application_name> --host <host>
--user <e-mail_or_user> --downtime-application <downtime_application_name>
```

Parameters

To list all parameters available for this command, execute **`neo help set-downtime-app`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`--downtime-application`	Downtime application name The downtime page application is provided by the customer and hosted in the same subaccount as the application itself. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Example

```
neo set-downtime-app --account mysubaccount --application myapp --user
mymail@example.com --downtime-application downtimeapp
```

Related Information

clear-downtime-app [page 1375]
Handle Unplanned Downtime [page 1617]

6.3.3.4.135 set-log-level

This command sets a log level for one or multiple loggers.

```
neo set-log-level  --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user> --host <host> --loggers
<logger_name1>,<logger_name2>,... --level <log_level>
```

Log Level Mapping

Simple Logging Facade for Java (SLF4J) uses the following log levels:

Level	Description
ALL	This level has the lowest possible rank and is intended to turn on all logging.
TRACE	This level designates finer-grained informational events than DEBUG.
DEBUG	This level designates fine-grained informational events that are most useful to debug an application.
INFO	This level designates informational messages that highlight the progress of the application at coarse-grained level.
WARN	This level designates potentially harmful situations.
ERROR	This level designates error events that might still allow the application to continue running.
OFF	This level has the highest possible rank and is intended to turn off logging.

> ⚠ Caution
>
> HTTP headers are logged in plain text once the log level is set to `DEBUG` or `ALL`. Thus, in case they contain any sensitive information, such as passwords kept in an HTTP **Authorization** header, it's disclosed in the logs.

Parameters

To list all parameters available for this command, execute **`neo help set-log-level`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-g, --loggers`	Single or multiple comma-separated logger names `Type`: string
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-l, --level`	The log level you want to set for the logger(s) `Type`: string
`-p, --password`	Password for the specified user. To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Your email, SAP ID or user name `Type`: string

Example

```
neo set-log-level --account mysubaccount --application demo --user p1234567890 --
host hana.ondemand.com --loggers com.acme.foo,com.acme.bar --level ERROR
```

Related Information

Using Logs in the Console Client
Exit Codes [page 1584]

6.3.3.4.136 set-quota

Sets compute unit quotas for a given subaccount.

> ### Note
>
> The amount you want to set cannot exceed the amount of quota you have purchased. In case you try to set bigger amount of quota, you will receive an error message.

```
neo set-quota --account <subaccount_technical_name> --host <host> --user <e-
mail_or_user> --amount <quota_type>:<integer_amount_of_quota>
```

Parameters

To list all parameters available for this command, execute **`neo help set-quota`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-m, --amount`	Compute unit quota type and amount of the quota to be set in the format <type>:[amount]. In this composite parameter, the <type> part is mandatory and must have one of the following values: lite, pro, prem, prem-plus. The amount part is optional and must be an integer value. If omitted, a default value 1 is assigned. Do not insert spaces between the two parts and their delimiter ":", and use lower case for the <type> part. `Type`: string

Example

```
neo set-quota --account mysubaccount --user myuser --host hana.ondemand.com --
amount lite:2
```

6.3.3.4.137 set-ssl-host

Configures and updates an SSL host. Allows you to replace an SSL certificate with a different one, manage TLS protocol versions, and configure a bundle of trusted CAs.

```
neo set-ssl-host --account <subaccount_technical_name> --user <e-mail_or_user> --
host <host> --name <ssl_hostname>
```

Parameters

To list all parameters available for this command, execute **`neo help set-ssl-host`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-n, --name`	Name of the SSL host that will be configured and updated.

Optional	
`-c, --certificate`	Name of the certificate that you bind to the SSL host. The certificate must already be uploaded. **⚠ Caution** This will replace the previously bound certificate, if there is already one. `Type`: string (It can contain alphanumerics, '.', '-', and '_')

Optional	
`--ca-bundle`	Use a switch to specify if client certificate authentication is mandatory for the respective CA bundle. For more information, see Managing Client Certificate Authentication for Custom Domains [page 1660]. `Type`: string `Format`: <bundle_name>:<switch> `Acceptable switch values`: • `request` - client certificate authentication is not mandatory • `require` - client certificate authentication is mandatory • `none` - removes the client certificate configuration for the specified bundle and unassigns that bundle from the SSL host `Default switch value`: request
`-t, --supported-protocols`	Specify the TLS protocols that you want to enable for the SSL host. The remaining TLS protocols are disabled. This parameter requires a certificate to be bound to the SSL host. `Type`: string `Supported TLS protocols`: TLSV1_2 Newer TLS versions will be added to the list of supported TLS protocols, if necessary. `Delimiter`: comma (,) **i Note** Enabling TLS 1.2 with `--supported-protocols TLSV1_2` disables all ciphers considered weak by the platform and listed in the --supported-ciphers section. You can enable these ciphers again using the`--supported-ciphers` parameter. To check the currently enabled TLS version, run the `set-ssl-host` command without using any optional parameters.

Optional

`-s, --supported-ciphers` — It allows you to enable additional ciphers for the SSL host.

> **i Note**
>
> This parameter does not work on its own. It is always accompanied by the `--supported-protocols` parameter with value `TLSV1_2`.

`Type`: string

`Acceptable values`:

- AES128_SHA256
- AES256_SHA256
- AES128_SHA
- AES256_SHA
- ECDHE_RSA_AES128_CBC_SHA
- ECDHE_RSA_AES128_SHA256
- ECDHE_RSA_AES256_CBC_SHA
- ECDHE_RSA_AES256_SHA384

`Delimiter`: comma (,)

> **⚠ Caution**
>
> For security reasons, it is recommended to use the default TLS 1.2 ciphers without using the `--supported-ciphers` parameter.

> **i Note**
>
> If you change either the supported protocols, ciphers, or both, you will set a custom SSL profile for which any further TLS versions will not be updated automatically.
>
> If you want to learn which TLS version and cipher your application is using to communicate with the Neo environment, see Transport Layer Security (TLS) Connectivity Support.

Examples

- ```
  neo set-ssl-host -a mysubaccount -u mymail@example.com -h hana.ondemand.com -n mysslhostname
  ```

  If the optional parameters are not used, the `set-ssl-host` command returns the current properties of the SSL host.

- ```
  neo set-ssl-host -a mysubaccount -u mymail@example.com -h hana.ondemand.com -n mysslhostname -c mycert --supported-protocols "TLSV1_2"
  ```

  Here, TLS 1.2 is enabled and all ciphers considered weak by the platform are disabled.

> **i Note**
>
> Enabling TLS 1.2 this way will disable the following ciphers: [ECDHE-RSA-AES128-CBC-SHA, ECDHE-RSA-AES128-SHA256, ECDHE-RSA-AES256-CBC-SHA, ECDHE-RSA-AES256-SHA384, AES128-SHA256, AES256-SHA256, AES128-SHA, AES256-SHA]. You can enable these ciphers again using the `--supported-ciphers` parameter.

- ```
  neo set-ssl-host -a mysubaccount -u mymail@example.com -h hana.ondemand.com -n mysslhostname -c mycert --supported-protocols "TLSV1_2" --supported-ciphers "AES128-SHA,AES256-SHA"
  ```

  In this example, AES128-SHA and AES256-SHA are the only ciphers enabled for TLS 1.2.

Related Information

create-ssl-host [page 1395]
upload-domain-certificate [page 1578]
bind-domain-certificate [page 1367]
change-domain-certificate [page 1371]
Bind the Certificate to the SSL Host [page 1654]
add-ca [page 1360]
list-cas [page 1483]
remove-ca [page 1522]

6.3.3.4.138 status

You can check the current status of an application or application process. The command lists all application processes with their IDs, state, last change date sorted chronologically, and runtime information.

The command also lists the availability zones where these application processes are running. However, this is only valid for recently started applications and if you have the latest SAP BTP SDK for Neo environment version installed.

The availability zones ensure the high availability of your application processes. If one of the availability zones experiences infrastructure issues and downtime, only the processes in this zone are affected. The remaining processes continue to run normally, ensuring that your application is working as expected.

When an application process is running but cannot receive new connection requests, it is marked as disabled in its status description. Additionally, if an application is in planned downtime and a maintenance page has been configured for it, the corresponding application is listed in the command output.

```
neo status --account <subaccount_technical_name> --application
<application_name> --host <host> --user <e-mail_or_user>
```

```
neo status --application-process-id <ID> --host <host> --user <e-mail_or_user>
```
```

Parameters

To list all parameters available for this command, execute **`neo help status`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) `Condition`: Not required if using `--application-process-id`
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) `Condition`: Not required if using `--application-process-id`
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-i, --application-process-id`	Unique ID of a single application process. Use it to show the status of a particular application process instead of the whole application. As the process ID is unique, you do not need to specify subaccount and application parameters. `Default`: none `Type`: string (hexadecimal sequence of 2 to 40 characters)
`--show-full-process-id`	Shows the full length (40 characters) of the unique application process ID. You may need to get the full ID when you try to execute a certain operation on the application process and the process cannot be identified uniquely with the short version of the ID. In particular, usage of the full length is recommended for tools and batch processing. If this parameter is not used, the status command lists only the first 7 characters by default. `Default`: off `Type`: switch, takes no value

Example

You can list all application processes in your application with their IDs:

```
neo status --host hana.ondemand.com --account mysubaccount --application myapp --
user mymail@example.com
```

Then, you can request the status of a particular application process from the list using its ID:

```
neo status --host hana.ondemand.com --application-process-id e8df21d --user
mymail@example.com
```

Related Information

Console Client for the Neo Environment [page 1348]
start [page 1557]
disable [page 1430]
start-maintenance [page 1560]

6.3.3.4.139 start

Starts a deployed application in order to make it available for customers. In case the application is already started, the command starts an additional application process if the quota for maximum allowed number of application processes is not exceeded.

```
neo start --account <subaccount_technical_name> --application <application_name>
--user <e-mail_or_user>
--host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help start`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required	
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Optional	
`--disabled`	Starts an application process in disabled state, so that it is not available for new connections. `Default`: off `Type`: switch, takes no value
`-y, --synchronous`	Triggers the starting process and waits until the application is started. The command without the `--synchronous` parameter triggers the starting process and exits immediately without waiting for the application to start. `Default`: off `Type`: switch, takes no value

Example

To start the application and wait for the operation to finish, execute:

```
neo start --host hana.ondemand.com --account mysubaccount --application myapp --
user mymail@example.com --synchronous
```

Related Information

Console Client for the Neo Environment [page 1348]
status [page 1555]
Scale Applications [page 1599]

6.3.3.4.140 start-db-hana

This command starts the specified SAP HANA tenant database on an SAP HANA tenant database (`MDC`) system.

```
neo start-db-hana -a <subaccount_technical_name> -h <host> -u <e-mail_or_user> -
i <database_ID>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-i, --id`	SAP HANA tenant database ID `Type`: string

Example

```
neo start-db-hana -a mysubaccount -h hana.ondemand.com -u mymail@example.com -i
mydb
```

Related Information

stop-db-hana [page 1563]
restart-hana [page 1529]

6.3.3.4.141 start-local

This command starts a local server instance.

```
neo start-local
```

Parameters

Optional	
`-l, --location`	Local server installation directory
`--shutdown-port`	Shutdown port opened by server `Default:` *8003*
`--wait-url`	Waits for a 2xx response from the specified URL before exiting
`--wait-url-timeout`	Seconds to wait for a 2xx response from the wait-url before exiting `Default:` *180*

Related Information

Deploy Locally with the Console Client [page 890]

6.3.3.4.142 start-maintenance

This command starts the planned downtime of an application, during which it no longer receives requests and a custom maintenance page for that application is shown to the user. All active connections will still be handled until the application is stopped.

```
neo start-maintenance --account <subaccount_technical_name> --application
<application_name> --host <host>
--user <e-mail_or_user> --maintenance-application <maintenance_application_name>
```

Parameters

To list all parameters available for this command, execute **`neo help start-maintenance`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`--maintenance-application`	Maintenance page application name The maintenance page application is provided by the customer and hosted in the same subaccount as the application itself. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Optional	
`--direct-access-code`	While setting your application in maintenance mode, you can generate an access code, which you can use later during the maintenance period. While your application is in maintenance mode, you can use this access code in the *Direct-Access-Code* HTTP header so that you can have access to your application for testing and administration purposes. In the meantime, users will continue to have access to the maintenance application. `Type`: string (alphanumeric; up to 100 characters)

If an application is already in planed downtime, executing the status command for it will show the maintenance application, to which the traffic is being redirected.

Example

```
neo start-maintenance --account mysubaccount --application myapp --user
<mymail@example.com --host hana.ondemand.com --maintenance-application maintapp
```

Related Information

Enable Maintenance Mode for Planned Downtimes [page 1613]
stop-maintenance [page 1565]
status [page 1555]

6.3.3.4.143 stop

Use this command to stop your deployed and started application or application process.

```
neo stop --account <subaccount_technical_name> --application <application_name>
--user <e-mail_or_user> --host <host>
```

```
neo stop --application-process-id <ID> --user <e-mail_or_user> --host <host>
```

Parameters

To list all parameters available for this command, execute **`neo help stop`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) `Condition`: Not required if using `--application-process-id`
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter) `Condition`: Not required if using `--application-process-id`
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-y, --synchronous`	Triggers the stopping process and waits until the application is stopped. The command without the `--synchronous` parameter triggers the stopping process and exits immediately without waiting for the application to stop. `Default`: off `Type`: switch, takes no value
`-i, --application-process-id`	Unique ID of a single application process. Use it to stop a particular application process instead of the whole application. As the process ID is unique, you do not need to specify subaccount and application parameters. You can list the application process ID by using the `<status>` command. `Default`: none `Type`: string (hexadecimal sequence of 2 to 40 characters)

Example

To stop the whole application and wait for the operation to finish, execute:

```
neo stop --host hana.ondemand.com --account mysubaccount --application myapp --
user mymail@example.com --synchronous
```

Related Information

Console Client for the Neo Environment [page 1348]
status [page 1555]
Exit Codes [page 1584]

6.3.3.4.144 stop-db-hana

This command stops the specified SAP HANA tenant database on an SAP HANA tenant database (`MDC`) system.

```
neo stop-db-hana -a <subaccount_technical_name> -h <host> -u <e-mail_or_user> -i
<database_ID>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-i, --id`	SAP HANA tenant database ID `Type`: string

Example

```
neo stop-db-hana -a mysubaccount -h hana.ondemand.com -u mymail@example.com -i
mydb
```

Related Information

start-db-hana [page 1559]
restart-hana [page 1529]

6.3.3.4.145 stop-local

This command stops a local server instance.

```
neo stop-local
```

Parameters

Optional	
`--shutdown-port`	Shutdown port opened by server `Default`: *8003*

Related Information

Deploy Locally with the Console Client [page 890]

6.3.3.4.146 stop-maintenance

This command stops the planned downtime of an application, starts traffic to it and deregisters the maintenance application page.

```
neo stop-maintenance --account <subaccount_technical_name> --application
<application_name> --host <host>
--user <e-mail_or_user>
```

Parameters

To list all parameters available for this command, execute **`neo help stop-maintenance`** in the command line.

Required	
`-a`, `--account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b`, `--application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h`, `--host`	Enter a region host. `Type`: URL, for acceptable values see Regions `Condition`: Do not specify if your host is https://hana.ondemand.com.

Required	
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

```
neo stop-maintenance --account mysubaccount --application myapp --user
<mymail@example.com
```

Related Information

Enable Maintenance Mode for Planned Downtimes [page 1613]
start-maintenance [page 1560]

6.3.3.4.147 subscribe

Subscribes the subaccount of the consumer to a provider Java application. Once the command is executed successfully, the subscription is visible in the *Subscriptions* panel of the cockpit in the consumer subaccount.

```
neo subscribe --account <subaccount_technical_name> --application
<provider_subaccount:application> --user <e-mail_or_user> --host <host>
```

> → Remember
>
> You must have the *Administrator* role in the provider and consumer subaccount to execute this command.

> i Note
>
> You can subscribe a subaccount to a Java application that is running in another subaccount only if both subaccounts (provider and consumer subaccount) belong to the same region.

Parameters

To list all parameters available for this command, execute **`neo help subscribe`** in the command line.

Required	
`-a, --account`	Consumer subaccount This is the subaccount of the consumer that is to be subscribed. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Provider subaccount and application This parameter must be specified in the format <provider subaccount >:<provider application>. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-u, --user`	Use your email, SAP ID or user name To be able to execute this command, the specified user must be a member of both the provider and the consumer subaccounts and must possess the Administrator role in those subaccounts. `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.

Example

```
neo subscribe --account consumersubaccount --application mysubaccount:myapp --
user myuser --host us1.hana.ondemand.com
```

Related Information

Subscribe to Java Multitenant Applications in the Neo Environment [page 838]
Providing Java Multitenant Applications to Tenants for Testing [page 836]
Subscribe to HTML5 Multitenant Applications in the Neo Environment [page 840]
unsubscribe [page 1577]
list-subscribed-accounts [page 1507]
list-subscribed-applications [page 1509]

6.3.3.4.148 subscribe-mta

This command subscribes the subaccount of the consumer to a Multitarget Application (MTA), which is available for subscription.

```
neo subscribe-mta --host <host> --account <subaccount_technical_name> --id
<provider_subaccount:mtaid> --user <e-mail_or_user>
```

Parameters

To list all parameters available for this command, execute **`neo help subscribe-mta`** in the command line.

Required	
`-a, --account`	The name of the subaccount for which you provide a user and a password.
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values see Regions
`-p, --password`	Your user password. We recommend that you enter it only when prompted, and not explicitly as a parameter in a properties file or the command line.
`-u, --user`	Your user e-mail or SAP ID (SCN) user name.
`--id`	Provider subaccount and a provided MTA This parameter must be specified in the format `<provider_subaccount>:<mtaid>`, where `<provider_subaccount>` is the subaccount that provides the MTA and `<mtaid>` is the ID of the provided MTA. Type: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Optional	
Command-specific parameters	
`-y, --synchronous`	Triggers the deployment and waits until the deployment operation finishes. The command without the `--synchronous` parameter triggers deployment and exits immediately without waiting for the operation to finish. Takes no value.
`-e, --extensions`	Defines one or more extensions to the deployment descriptor. A comma-separated list of file locations, pointing to the extension descriptor files, or the folders containing them. For more information, see Defining MTA Extension Descriptors.

Example

```
neo subscribe-mta --host us1.hana.ondemand.com --account mysubaccount --id
providedsubaccount:com.sap.example.mta --user mymail@example.com
```

6.3.3.4.149 unbind-db

This command unbinds a database from a Java application for a particular data source.

The application retains access to the database until the next application restart. After the restart, the application will no longer be able to access it.

```
neo unbind-db -a <subaccount_technical_name> -b <application_name> -h <host> -u
<e-mail_or_user>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Optional	
`-s, --data-source`	Data source name Default: <DEFAULT>

Example

```
neo unbind-db -a mysubaccount -b myapp -h hana.ondemand.com -u mymail@example.com
```

Related Information

bind-db [page 1364]
Binding SAP HANA Databases to Java Applications
Binding SAP ASE Databases to Java Applications
Binding SAP HANA Databases to Java Applications
Bind Applications to Databases in the Same Global Account
Bind Applications to Databases in Other Subaccounts

6.3.3.4.150 unbind-domain-certificate

Unbinds a certificate from an SSL host. The certificate will not be deleted from SAP BTP storage.

```
neo unbind-domain-certificate --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --ssl-host <ssl_hostname>
```

Parameters

To list all parameters available for this command, execute **`neo help unbind-domain-certificate`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Required	
`-l, --ssl-host`	SSL host as defined with the `--name` parameter when created, or 'default' if not specified.

Example

```
neo unbind-domain-certificate --account mysubaccount --user mymail@example.com --
host hana.ondemand.com --ssl-host mysslhostname
```

Related Information

Bind the Certificate to the SSL Host [page 1654]
Update an Expired Certificate [page 1660]

6.3.3.4.151 unbind-hana-dbms

This command unbinds a productive SAP HANA database system from a Java application for a particular data source.

The application retains access to the productive SAP HANA database system until the next application restart. After the restart, the application will no longer be able to access the database system.

```
neo unbind-hana-dbms -a <subaccount_technical_name> -b <application_name> -h
<host> -u <e-mail_or_user>
```

Parameters

Required	
`-a, --account`	Subccount name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required	
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Optional	
`-s, --data-source`	Data source name

Example

```
neo unbind-hana-dbms -a mysubaccount -b myapp -h hana.ondemand.com -u
mymail@example.com
```

Related Information

bind-hana-dbms [page 1368]
Binding SAP HANA Databases to Java Applications
Bind Applications to Databases in the Same Global Account
Bind Applications to Databases in Other Subaccounts

6.3.3.4.152 unbind-schema

This command unbinds a schema from an application for a particular data source.

The application retains access to the schema until the next application restart. After the restart, the application will no longer be able to access the schema.

```
neo unbind-schema -a <subaccount_technical_name> -b <application_name> -h <host>
-u <e-mail_or_user>
```

Parameters

Required	
`-a, --account`	Subccount name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your e-mail, SAP ID, or user name `Type`: string

Optional	
`-s, --data-source`	Data source name

Example

```
neo unbind-schema -a mysubaccount -b myapp -h hanatrial.ondemand.com -u
mymail@example.com -s datasource1
```

Related Information

Example Scenarios
Administering Database Schemas
bind-schema [page 1370]

6.3.3.4.153 undeploy

Undeploying an application removes it from SAP BTP. To undeploy an application, you have to stop it first.

If you use SAP Enhanced Disaster Recovery service, the application is undeployed first on the disaster recovery region and then on the specified region.

```
neo stop --host <host> --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user>
```

```
neo undeploy --host <host> --account <subaccount_technical_name> --application
<application_name> --user <e-mail_or_user>
```

Parameters

To list all parameters available for this command, execute the **`neo help undeploy`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Application name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Example

First stop and then undeploy the application.

```
neo stop --host hana.ondemand.com --account mysubaccount --application myapp --
user mymail@example.com
```

```
neo undeploy --host hana.ondemand.com --account mysubaccount --application myapp
--user mymail@example.com
```

Related Information

Console Client for the Neo Environment [page 1348]
stop [page 1562]
Exit Codes [page 1584]

6.3.3.4.154 unmap-proxy-host

Deletes the mapping between an application host and an on-premise reverse proxy host and port.

```
neo unmap-proxy-host --account <subaccount_technical_name> --app-host
<application_host> --proxy <proxy_host:port> -h <host> -u <e-mail_or_user> -p
<password>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`--app-host`	Your application hostname. For example: `myapp.hana.ondemand.com`
`--proxy`	On-premise reverse proxy hostname and port. Separate proxy hostname and port with a colon (':'). For example: `loc.corp:123`
`-h, --host`	Region host on which you execute the command. `Type`: URL. For acceptable values see Regions.
`-u, --user`	Your email, SAP ID, or username. `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string

Example

```
neo unmap-proxy-host --account mysubaccount --app-host app.hana.ondemand.com --
proxy loc.corp:123 -h hana.ondemand.com -u username -p ******
```

An application with hostname `app.hana.ondemand.com` is unmapped from proxy host `loc.corp:123`.

Related Information

Configuring Application Access via On-Premise Reverse Proxy [page 1668]
map-proxy-host [page 1513]
list-proxy-host-mappings [page 1499]

6.3.3.4.155 unregister-access-point

Unregisters the access point URL registered for a virtual machine specified by name or ID.

```
neo unregister-access-point --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --name <vm_name>
```

Parameters

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-i , --id`	ID of the virtual machine `Type`: string `Condition`: Use either `--id` or `--name`
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-n, --name`	Name of the virtual machine `Type`: string `Condition`: Use either `--id` or `--name`
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string

Example

```
neo unregister-access-point --account mysubaccount --user mymail@example.com --
host hana.ondemand.com --name myvm
```

Related Information

register-access-point [page 1521]
Enable Internet Access

6.3.3.4.156 unsubscribe

Removes the subscription to a provider Java application from a consumer subaccount.

```
neo unsubscribe --account <subaccount_technical_name> --application
<provider_subaccount:application> --user <e-mail_or_user> --host <host>
```

> → Remember
>
> You must have the Administrator role in the provider and consumer subaccount to execute this command.

Parameters

To list all parameters available for this command, execute **`neo help unsubscribe`** in the command line.

Required	
`-a, --account`	Subaccount technical name This is the subaccount of the consumer that is to be unsubscribed. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-b, --application`	Name of provider subaccount and application name This parameter must be specified in the format <subaccount name>:<application>. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)

Required	
`-u, --user`	Use your email, SAP ID or user name To be able to execute this command, the specified user must be a member of both the provider and the consumer subaccounts. `Type`: string
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.

Example

```
neo unsubscribe --account consumersubaccount --application mysubaccount:myapp --
user myuser --host us1.hana.ondemand.com
```

Related Information

Subscribe to Java Multitenant Applications in the Neo Environment [page 838]
Subscribe to HTML5 Multitenant Applications in the Neo Environment [page 840]
subscribe [page 1566]
list-subscribed-accounts [page 1507]
list-subscribed-applications [page 1509]

6.3.3.4.157 upload-domain-certificate

Uploads a signed custom domain certificate to SAP BTP. You can upload either a certificate based on a previously generated CSR via the `generate-csr` command, or another valid certificate with its corresponding private key.

```
neo upload-domain-certificate --account <subaccount_technical_name> --user <e-
mail_or_user> --host <host> --name <certificate_name> --location <file_location>
```

Parameters

To list all parameters available for this command, execute **`neo help upload-domain-certificate`** in the command line.

Required	
`-a, --account`	Subaccount technical name `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL. For acceptable values, see Regions.
`-p, --password`	To protect your password, enter it only when prompted by the console client and not explicitly as a parameter in the properties file or the command line. `Type`: string
`-u, --user`	Use your email, SAP ID, or user name. `Type`: string
`-n, --name`	Name of the certificate previously used in the CSR generation via the `generate-csr` command. If you upload a certificate not based on a CSR generated via `generate-csr`, you use this parameter to name the certificate. `Type`: string The certificate name must start with a letter and can only contain lowercase letters (a-z), uppercase letters (A-Z), numbers (0-9), underscores (_), and hyphens (-).
`-l, --location`	Location of a file containing certificate data **i Note** Some CAs issue chained root certificates that contain one or more intermediate certificates. In such cases, put all certificates in the file for upload starting with the signed SSL certificate. **⚠ Caution** Once uploaded, the certificate cannot be downloaded for security reasons. This also includes intermediate certificates. `Type`: file location `Maximum file size`: 20 KB

Optional	
`-f, --force`	Overwrites an existing SSL certificate. For example, this parameter lets you update an expired certificate based on an already existing CSR. For more information, see Using the CSR of the Bound Certificate [page 1662]. The `--force` option is also useful if you had to and you did not upload an intermediate certificate for some reason. Note that the intermediate certificate must be added to the file that contains the SSL certificate.
`-k, --key-location`	Location of the file containing the private key of the certificate specified in `--name` If you want to upload a signed certificate that is not based on a CSR generated via the `generate-csr` command, you must use this parameter to remotely upload this certificate to SAP BTP along with its private key. **⚠ Caution** Uploading a private key from a remote location poses a security risk. Also, there is no way to download the uploaded private key. SAP recommends that you use only certificates that are based on CSRs previously generated via the `generate-csr` command. `Type`: file location `Maximum file size`: 4 KB

Examples

```
neo upload-domain-certificate --account mysubaccount --user mymail@example.com --
host hana.ondemand.com --name myfirstcert --location ./certificate.pub
```

The `.pub` file (for example, certificate.pub) should contain either:

- An SSL certificate.

  Example

  ```
  -----BEGIN CERTIFICATE-----
  Enter your SSL certificate.
  The certificate must be in Privacy-enhanced Electronic Mail (PEM) format
  (128 or 256 bits).
  -----END CERTIFICATE-----
  ```

- An SSL certificate followed by an intermediate certificate.

  Example

  ```
  -----BEGIN CERTIFICATE-----
  Enter your SSL certificate.
  The certificate must be in Privacy-enhanced Electronic Mail (PEM) format
  (128 or 256 bits).
  -----END CERTIFICATE-----
  -----BEGIN CERTIFICATE-----
  ```

```
Enter your intermediate certificate.
The certificate must be in Privacy-enhanced Electronic Mail (PEM) format
(128 or 256 bits).
-----END CERTIFICATE-----
```

```
neo upload-domain-certificate --account mysubaccount --user mymail@example.com --
host hana.ondemand.com --name mycert --location ./certificate.pub --key-
location ./certificate.key
```

It is not recommended, but if you decide to upload a private key (for example, certificate.key), it should follow one of these formats:

Example

```
-----BEGIN PRIVATE KEY-----
Enter your private key.
It must be without password, in Privacy-enhanced Electronic Mail (PEM) format
(2048-4096 bits).
-----END PRIVATE KEY-----
```

Example

```
-----BEGIN RSA PRIVATE KEY-----
Enter your RSA private key.
It must be without password, in Privacy-enhanced Electronic Mail (PEM) format
(2048-4096 bits).
-----END RSA PRIVATE KEY-----
```

Related Information

generate-csr [page 1446]
Configuring Custom Domains [page 1649]

6.3.3.4.158 upload-keystore

This command is used to upload a keystore by uploading the keystore file. You can upload keystores on subaccount, application, and subscription levels.

Parameters

To list all parameters available for this command, execute **`neo help upload-keystore`** in the command line.

Required	
`-a, --account`	Consumer subaccount technical name The account for which you provide username and password. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-h, --host`	Enter a region host. `Type`: URL, for acceptable values see Regions.
`-l,--location`	Path to a keystore file to be uploaded from the local file system. The file extension determines the keystore type. The following extensions are supported: `.jks`, `.jceks`, `.p12`, `.pem`. For more information about the keystore formats, see Features [page 1782] `Type`: string
`-u, --user`	Use your email, SAP ID or user name `Type`: string

Optional	
`-b, --application`	Application name • Use `--application <consumer_application_name>` if the application is running in your subaccount. • Use `--application <provider_subaccount_technical_name>:<provider_application_name>` if the application is running in another subaccount. `Type`: string (up to 30 characters; lowercase letters and numbers, starting with a letter)
`-w, --overwrite`	Overwrites a file with the same name if such already exists. If you do not explicitly include the `--overwrite` argument, you will be notified and asked if you want to overwrite the file.

Example

On Subscription Level

```
neo upload-keystore --account <consumer_subaccount_technical_name> --application
<provider_subaccount_technical_name>:<provider_application_name>
--user <e-mail_or_user> --location C:\Keystores\KeyStore1.jks --host
hana.ondemand.com
```

On Application Level

```
neo upload-keystore --account <provider_subaccount_technical_name> --application
<application_name>
--user <e-mail_or_user> --location C:\Keystores\KeyStore1.jks --host
hana.ondemand.com
```

On Subaccount Level

```
neo upload-keystore --account <subaccount_technical_name> --user <e-mail_or_user>
--location C:\Keystores\KeyStore1.jks --host hana.ondemand.com
```

Related Information

Keystore Console Commands [page 1783]
Keys and Certificates [page 1781]
Using the Keystore Service for Client Side HTTPS Connections [page 1791]

6.3.3.4.159 version

This command is used to list the SDK for SAP BTP,Neo environment version and the runtime. It also lists the command versions and the JAR files in the SDK and checks whether the SDK is up to date.

Use this command to show the SDK for SAP BTP, Neo environment version and the runtime. You can use parameters to list the command versions and the JAR files in the SDK and to check whether the SDK version is up to date.

```
neo version --commands
```

```
neo version --jars
```

```
neo version --updates
```

Parameters

To list all parameters available for this command, execute `neo help` version in the command line.

Required	
`-c, --commands`	Lists all commands available in the SDK and their versions.
`-j, --jars`	Lists all JAR files in the SDK and their versions.
`-u, --updates`	Checks if there are any updates and hot fixes for the SDK and whether the SDK version is still supported. It also provides the version of the latest available SDK.

Optional	
`--output <value>`	Prints the output in the specified format. `Acceptable values`: 'json' `Type`: string

Example

To show the SAP BTP SDK for Neo environment version and the runtime, execute:

```
neo version
```

To list all available commands and their versions, execute:

```
neo version -c
```

To list all JAR files in the SDK and their versions, execute:

```
neo version -j
```

To check whether the SDK is up to date, execute:

```
neo version -u
```

There are several possible outcomes:

- a hot fix is available, you need to update your SDK
- your SDK is the latest version available
- your SDK is deprecated, you need to update it
- your SDK is supported, but it is not the latest version available

Related Information

Machine-Readable Command Output [page 1353]

6.3.3.5 Exit Codes

Overview

The exit code is a number that indicates the outcome of a command execution. It shows whether the command completes successfully or defines an error if something goes wrong during the execution.

When commands are executed as part of automated scripts, the exit codes provide feedback to the scripts, which allows the script to bypass known errors that can be met during execution. A script can also interact with the user in order to request additional information required for the script to complete.

All exit codes in SAP BTP are aligned to the Bash-Scripting Guide. For more information, see Exit Codes With Special Meanings .

Ranges

The set of exit codes is divided into ranges, based on the error type and the reason.

Error Type	Start Number	End Number	Count
No error	0	0	1
Common errors	1	9	9
Missing parameters	10	39	30
Parameter validation errors	40	109	70
Authentication and Authorization Errors	110	126	17
Reserved space for system errors	127	165	39
Command-specific errors: frontend	166	209	44
Command-specific errors: backend	210	254	45
Reserved space for system errors	255	255	1

Exit Codes

Exit codes can be defined as *general* (common for all commands) and *command-specific* (cover different cases via different commands).

Code	Meaning	Type/Reason
0	OK	
1	General error	Error during execution of command

Code	Meaning	Type/Reason
2	Command not found	Misspelled or missing command
3	Unsupported/Incompatible SDK version	SDK is not compatible with the runtime
4	Network error	Network problem or missing proxy configuration
5-9	Currently not used	Common errors
10	General missing parameter	Missing parameters
11	Missing host name	Missing parameters
12	Missing account name	Missing parameters
13	Missing application name	Missing parameters
14	Missing user name	Missing parameters
15-19	Currently not used	Missing parameters
20-39	Available for use by commands	Missing parameters
40	General parameter is invalid or empty	Validation errors
41	Host name parameter is invalid or empty	Validation errors
42	Account name parameter is invalid or empty	Validation errors
43	Application name parameter is invalid or empty	Validation errors
44-49	Currently not used	Validation errors
50-109	Available for use by commands	Validation errors
110	Wrong user or password	Authentication and authorization errors
111	General authentication and authorization error	Authentication and authorization errors
112-114	Currently not used	Authentication and authorization errors
115-126	Available for use by commands	Authentication and authorization errors
127-165	Special exit codes	System-dependent errors
166	General frontend error	Frontend
167 - 209	Available for use by commands	Frontend
210	General backend error	Backend
211 - 254	Available for use by commands	Backend
255	Special exit codes	System-dependent errors

Related Information

Console Client for the Neo Environment

6.4 Application Operations in the Neo Environment

To learn about	See
How to configure and operate your deployed Java applications.	Java: Application Operations [page 1589]
How to monitor your SAP HANA applications.	SAP HANA: Application Operations [page 1631]
How to monitor the current status of the HTML5 applications in your subaccount.	HTML5: Application Operations [page 1631]
How to securely operate and monitor your cloud applications connected to on-premise systems.	Security
How to change the default SAP BTP application URL by configuring custom or platform domains.	Configuring Application URLs [page 1644]
How to enable transport of SAP BTP applications via the CTS+.	Transporting Multitarget Applications with CTS+ [page 1056]

6.4.1 Using Multiple Subaccounts for Staged Application Development

SAP BTP allows you to achieve isolation between the different application life cycle stages (development, testing, productive) by using multiple subaccounts.

Prerequisites

- You have developed an application. For more information, see Developing Java Applications [page 871].
- You have a subaccount in an enterprise account. For more information, see Global Accounts.

Context

Using multiple subaccounts ensures better stability. Also, you can achieve better security for productive applications because permissions are given per subaccount.

For example, you can create three different subaccounts for one application and assign the necessary amount of compute unit quota to them:

- **dev** - use for development purposes and for testing the increments in the cloud, you can grant permissions to all application developers

- **test** - use for testing the developed application and its critical configurations to ensure quality delivery (integration testing and testing in productive-like environment prior to making it publicly available)
- **prod** - use to run productive applications, give permissions only to operators.

You can create multiple subaccounts and assign quota to them either using the console client or the cockpit.

Create Multiple Subaccounts from the Cockpit

Procedure

1. Open the cockpit and navigate to your global account.
2. Choose *New Subaccount* and specify a display name.
3. Use the "+" and "-" buttons next to quota types to set the desired amount of quota and choose the *Save* button.

   Next, you can deploy your application in the newly created subaccount using the Eclipse IDE or the console client. Then, you can test your application and make it ready for productive use.

   You can transfer the application from one subaccount to another by redeploying it in the respective subaccount.

Create Multiple Subaccounts with the Console Client

Procedure

1. Open the command prompt and navigate to the folder containing `neo.bat/sh (<SDK installation folder>/tools)`.
2. Create a new subaccount.

   Execute:

   ```
   neo create-account -a <subaccount> -u <user who is member of that subaccount> -h <host> -n <display name of the new subaccount>
   ```
3. Assign the necessary amount of quota to the subaccount.

   Execute:

   ```
   neo set-quota -a <subaccount> -u <user name or email> -h <host> -m <type of the quota lite, pro, prem or prem-plus>:<integer amount of the quota>
   ```

   Next, you can deploy your application in the newly created subaccount by executing **`neo deploy -a <subaccount> -h <host> -b <application name> -s <file location> -u <user name or email>`**. Then, you can test your application and make it ready for productive use.

   You can transfer the application from one subaccount to another by redeploying it in the respective subaccount.

Related Information

Org Administration Using the Cockpit
Deploying and Updating Applications [page 875]
create-account [page 1379]
delete-account [page 1400]
list-accounts [page 1478]
set-quota [page 1550]

6.4.2 Java: Application Operations

After you have developed and deployed your Java application on SAP BTP, you can configure and operate it using the cockpit, the console client, or the Eclipse IDE.

Content
Configuring Applications [page 1589]
Managing the Lifecycle of Deployed Applications [page 1590]
Monitoring [page 1591]
Profiling [page 1591]
Logging [page 1591]
Updating Productive Applications [page 1591]

Configuring Applications

Cockpit	View Subaccount Usage Analytics [page 1288] View Runtime Information [page 1608]

Console Client	Update Application Properties [page 1593] Choose Application Runtime Version [page 1594] Choose JRE Version [page 1595] Enable and Configure Gzip Response Compression [page 1597] Configure VM Arguments [page 1598] Scale Applications [page 1599]	Specify various configurations using commands.
Eclipse IDE		Use the options for advanced server and application configurations as well as direct reference to the cockpit UI.

Managing the Lifecycle of Deployed Applications

Cockpit	Define Application Details (Java Apps) [page 1602] Start and Stop Applications [page 1602] Check the Process Status [page 1605] View Subaccount Usage Analytics [page 1288]	Start, stop, and undeploy applications, as well as start, stop, and disable individual application processes.
Console Client	start [page 1557]; stop [page 1562]; restart [page 1528] enable [page 1443]; disable [page 1430]; undeploy [page 1574]	Manage the lifecycle of a deployed application or individual application processes by executing the respective command.
Lifecycle Management API	Start an Application [page 887] Stop an Application [page 889]	Start and stop applications using the Lifecycle Management API.

Monitoring

Cockpit, Console Client, Monitoring API	Monitoring Java Applications [page 729] Monitoring HTML5 Applications [page 753] Monitoring Database Systems [page 760]	View the current metrics or the metrics history. Configure checks for an application. Use the Metrics REST API to get the state or the metric details of an application.

Profiling

Eclipse IDE	Profiling Applications [page 1618]	Analyze resource-related problems in your application.

Logging

Cockpit	Using Logs in the Cockpit for Java Applications	View the logs and change the log settings of any applications deployed in your subaccount.
Console Client	Using Logs in the Console Client	Manage some of the logging configurations of a started application.

Updating Productive Applications

Cockpit	Enable Maintenance Mode for Planned Downtimes [page 1613] Perform Soft Shutdown [page 1615]	Supports zero downtime and planned downtime scenarios. Disable the application or individual processes in order to shut down the application or processes gracefully.
Console Client	Update Applications with Zero Downtime [page 1610] Enable Maintenance Mode for Planned Downtimes [page 1613] Perform Soft Shutdown [page 1615]	Deploy a new version of a productive application or perform maintenance.

6.4.2.1 Configuring Applications

As an operator, you can configure an SAP BTP application according to your scenario.

Make specific application configurations during deploy

When you are deploying the application using SAP BTP console client, you can specify various configurations using the deploy command parameters:

Choose Application Runtime Version [page 1594]

Enable and Configure Gzip Response Compression [page 1597]

Configure VM Arguments [page 1598]

Choose JRE Version [page 1595]

Update application properties

You can change the properties you specified during deployment.

Update Application Properties [page 1593]

Scale your application

You can scale an application to ensure its ability to handle more requests.

Scale Applications [page 1599]

Configure identity and access management

Using the cockpit, you can perform the following identity and access management configuration tasks:

Managing Roles [page 1710]

Application Identity Provider [page 1720]

OAuth 2.0 Configuration [page 1760]

Configure connectivity destinations

Using the cockpit and the console client, you can configure HTTP, Mail and RFC destinations to make use of them in your applications:

Configure Destinations from the Cockpit [page 77]

Configure Destinations from the Console Client [page 56]

View and download log files

Using the cockpit and the console client, you can view and download log files of any applications deployed in your subaccount:

Using Logs in the Cockpit for Java Applications

Using Logs in the Console Client

Related Information

Console Client for the Neo Environment [page 1348]

6.4.2.1.1 Update Application Properties

You can update a property of an application running on SAP BTP without redeploying it.

Context

Application properties are configured during deployment with a set of deploy parameters in the SAP BTP console client. If you want to change any of these properties (Java version, runtime version, compression, VM arguments, compute unit size, URI encoding, minimum and maximum application processes) without the need to redeploy the application binaries, use the `set-application-property` command. Execute the command separately for each property that you want to set.

Procedure

1. Open the command prompt and navigate to the folder containing neo.bat/sh (<SDK installation folder>/tools).

2. Execute `set-application-property` specifying the new value of one property that you want to change. For example, to change the compute unit size to premium, execute:

```
neo set-application-property myapp.properties --compute-unit prem
```

3. For the change to take effect, restart your application using the `restart` command.

Related Information

set-application-property [page 1541]
deploy [page 1422]
restart [page 1528]
Deploy on the Cloud with the Cockpit [page 894]

6.4.2.1.2 Choose Application Runtime Version

Applications deployed on SAP BTP are always started on the latest version of the application runtime container. This version contains all released fixes, critical patches and enhancements and is respectively the recommended option for applications. In some special cases, you can choose the version of the runtime container your application uses by specifying it with the parameter `<--runtime-version>` when deploying your application. To change this version, you need to redeploy the application without specifying this parameter.

Prerequisites

You have downloaded and configured SAP BTP console client. For more information, see Set Up the Console Client [page 847].

Context

If you want to choose the version of the application runtime container, follow the procedure.

Procedure

1. Open the command prompt and navigate to the folder containing neo.bat/sh (<SDK installation folder>/tools).
2. In the console client command line, execute the `<list-runtime-versions>` command to display all recommended versions. We recommend that you choose the latest available version.

3. Redeploy your application with parameter <--runtime-version> set to the selected version number.

```
neo deploy --account <subaccount_name> --application <application_name> --
source <file_location> --user <email_or_user>
 --runtime-version <your_chosen_version >
```

4. Restart your application using the <restart> command.

   If you want to return the default behavior when the application is always started on the latest version of the application runtime, redeploy your application without specifying the <--runtime-version> parameter.

> ⚠ Caution
>
> By selecting an older version of the application runtime, you do not have the latest released fixes and critical patches as well as enhancements, which may affect the smooth operation and supportability of your application. Consider updating the selected version periodically. Plan the updates to the latest version of the application runtime and apply in your test environment first. Older application runtime versions will be deprecated and expire. Refer to the <list-runtime-versions> command for information.

Related Information

deploy [page 1422]
start [page 1557]
View Runtime Information [page 1608]

6.4.2.1.3 Choose JRE Version

You can choose the Java Runtime Environment (JRE) version used for an application.

Prerequisites

You have downloaded and configured SAP BTP console client.

For more information, see Set Up the Console Client [page 847].

Context

The JRE version depends on the type of the SAP BTP SDK for Neo environment you are using. By default the version is:

- SDK for Java Web (1.x) - JRE 7

- SDK for Java EE 6 Web Profile (2.x) - JRE 7
- SDK for Java Web Tomcat 7 (2.x) – JRE 7
- SDK for Java Web Tomcat 8 (3.x) – JRE 8
- SDK for Java EE 7 Web Profile TomEE 7 (1.x) – JRE 8

If you want to change this default version, you need to specify the *--java-version* parameter when deploying the application using the SAP BTP console client. Only the version number of the JVM can be specified.

You can use JRE 8 with the Java Web Tomcat 7 runtime (neo-java-web version 2.25 or higher) in productive accounts.

For applications developed using the SAP BTP SDK for Neo environment for Java Web Tomcat 7 (2.x), the default JRE is 7. If you are developing a JSP application using JRE 8, you need to add a configuration in the `web.xml` that sets the compiler target VM and compiler source VM versions to 1.8.

Procedure

1. Open the command prompt and navigate to the folder containing neo.bat/sh (`<SDK installation folder>/tools`).
2. Deploy the application specifying *--java-version*. For example, to use JRE 7, execute the following command:

   ```
   neo deploy --account <subaccount_name> --application <application_name> --
   source <file_location>
   --user <e-mail_or_user> --java-version 7
   ```

   For Java Web Tomcat 8, Java version 8 is supported by default, but you can also use Java version 7.

Related Information

deploy [page 1422]
Org Administration Using the Cockpit

6.4.2.1.4 Enable and Configure Gzip Response Compression

Usage of gzip response compression can optimize the response time and improve interaction with an application as it reduces the traffic between the Web server and browsers. Enabling compression configures the server to return zipped content for the specified MIME type and size of the response.

Prerequisites

You have downloaded and configured SAP BTP console client.

For more information, see Set Up the Console Client [page 847].

Context

You can enable and configure gzip using some optional parameters of the deploy command in the console client. When deploying the application, specify the following parameters:

Procedure

1. To enable gzip compression, specify *--compression on*.
2. To configure response MIME types that will be compressed, use *--compressible-mime-type*.
3. To specify the size of responses which will be compressed, use *--compression-min-size*.

   If you enable compression but do not specify values for *--compressible-mime-type* or *--compression-min-size*, then the defaults are used: *text/html, text/xml, text/plain* and *2048* bytes, respectively.

   If you specify values for *--compressible-mime-type* or *--compression-min-size* but do not enable compression, then the operation passes, compression is not enabled and you get a warning message.

   If you want to enable compression for all responses independently from MIME type and size, use only *--compression force*.

Example

```
neo deploy myapp.properties --compression on --compressible-mime-type
application/javascript,application/json
 --compression-min-size 1024
```

Next Steps

Once enabled, you can disable the compression by redeploying the application without the compression options or with parameter *--compression off*.

Related Information

Console Client for the Neo Environment [page 1348]
deploy [page 1422]

6.4.2.1.5 Configure VM Arguments

Using SAP BTP console client, you can configure the JRE by specifying custom VM arguments.

Prerequisites

You have downloaded and configured the console client.

For more information, see Set Up the Console Client [page 847].

Context

You can configure the following arguments:

- System properties - they will be used when starting the application process. For example *{{-D<key>=<value>}}*
- Memory arguments - use them to define custom memory settings of your compute units. The supported memory settings are:
  -Xms<size> - set initial Java heap size
  -Xmx<size> - set maximum Java heap size
  -XX:PermSize - set initial Java Permanent Generation size
  -XX:MaxPermSize - set maximum Java Permanent Generation size

> **i Note**
>
> We recommend that you use the default memory settings. Change them only if necessary and note that this may impact the application performance or its ability to start.

Procedure

1. Open the command prompt and navigate to the folder containing neo.bat/sh (`<SDK installation folder>/tools`).
2. Deploy the application, specifying your desired configurations. For example, if you want to specify a currency and maximum heap size 1 GiB, then execute the deploy with the following parameters:

```
neo deploy myapp.properties --vm-arguments "-Dcurrency=EUR -Xmx1024m"
```

> **i Note**
>
> If you are deploying using the properties file, note that you have to use double quotation marks twice: **`vm-arguments=""-Dcurrency=EUR -Xmx1024m""`**.

This will set the system properties *-Dcurrency=EUR* and the memory argument *-Xmx1024m*.

To specify a value that contains spaces (for example, *-Dname=John Doe*), note that you have to use single quotation marks for this parameter when deploying.

```
neo deploy myapp.properties --vm-arguments "-Dcurrency=EUR '-Dname=John Doe' -Xmx1024m"
```

Related Information

Console Client for the Neo Environment [page 1348]
deploy [page 1422]

6.4.2.1.6 Scale Applications

Each application is started on a dedicated SAP BTP Runtime. One application can be started on one or many application processes, according to the compute unit quota that you have.

Prerequisites

- You have downloaded and configured SAP BTP console client. For more information, see Set Up the Console Client [page 847].
- Your application can run on more than one application processes

Context

Scaling an application ensures its ability to handle more requests, if necessary. Scalability also provides failover capabilities - if one application process crashes, the application will continue to work. First, when deploying the application, you need to define the minimum and maximum number of application processes. Then, you can scale the application up and down by starting and stopping additional application processes. In addition, you can also choose the compute unit size, which provides a certain central processing unit (CPU), main memory and disk space.

Procedure

1. Open the command prompt and navigate to the folder containing neo.bat/sh (`<SDK installation folder>/tools`).
2. Deploy the application, specifying *--minimum-processes* and *--maximum-processes*. The *--minimum-processes* parameter defines the number of processes on which the application is started initially. Make sure it is at least 2.

   ```
   neo deploy myapp.properties --minimum-processes 2 --maximum-processes 5
   ```

3. Start the application. It will be started on 2 application processes.

   ```
   neo start myapp.properties
   ```

4. You can now scale the application up by executing the start command again. Each new execution starts another application process. You can repeat the start until you reach the maximum number of application process you defined within the quota you have purchased.

   ```
   neo start myapp.properties
   ```

5. If for some reason you need to scale the application down, you can stop individual application processes by using soft shutdown. Each application process has a unique process ID that you can use to disable and stop the process.
   a. List all application processes with their attributes (ID, status, last change date) by executing *neo status* and identify the application process you want to stop.

      ```
      neo status myapp.properties
      ```

   b. Execute *neo disable* for the application process you want to stop.

      ```
      neo disable myapp.properties --application-process-id <ID>
      ```

   c. Wait for some time so that all working sessions finish.
   d. Stop the application process by executing *neo stop* with the appropriate parameters.

      ```
      neo stop myapp.properties --application-process-id <ID>
      ```

Next Steps

You can also scale your application vertically by choosing the compute unit size on which it will run after the deploy. You can choose the compute unit size by specifying the `--size` parameter when deploying the application.

For example, if you have an enterprise account and have purchased a package with Premium edition compute units, then you can run your application on a Premium compute unit size, by executing

```
neo deploy --size prem myapp.properties
```

Related Information

Compute Units [page 867]
deploy [page 1422]
status [page 1555]
Perform Soft Shutdown [page 1615]

6.4.2.2 Managing Deployed Applications

For an overview of the current status of the individual applications in your subaccount, use the cockpit. It provides key information in a summarized form and allows you to initiate actions, such as starting, stopping, and undeploying applications.

Related Information

Define Application Details (Java Apps) [page 1602]
Start and Stop Applications [page 1602]
Check the Process Status [page 1605]
View Monitoring Metrics of a Java Application [page 730]
Default Trace File [page 1606]
View Runtime Information [page 1608]
Using Logs in the Cockpit for Java Applications
Console Client Commands [page 1355]

6.4.2.2.1 Define Application Details (Java Apps)

You can view details about your currently selected Java application. By adding a suitable display name and a description, you can identify the application more easily.

Context

In the overview of a Java application in the cockpit, you can add and edit the display name and description for the Java application as needed.

The following details are available:

- Display name - a human-readable name that you can specify for your Java application and change it later on, if necessary.
- Description - a short descriptive text about the Java application, typically stating what it does.

Procedure

1. Log on to the cockpit and select a subaccount.
2. Choose *Java Applications* in the navigation area.
3. In the application list, select your application to go to the overview.
4. In the *Application Details* overview, choose *Edit*.
5. Specify or modify the display name or the description as needed and save your changes.

6.4.2.2.2 Start and Stop Applications

You can directly start, stop, and undeploy applications, as well as start, stop, and disable individual application processes.

Context

An application can run on one or more application processes. The use of multiple processes allows you to distribute application load and provide failover capability. The number of processes that you can start depends on the compute unit quota available to your global account and how an individual application has been configured. If you reach the maximum, increase the maximum number of processes first before you can start another process.

> **i Note**
>
> By default the application is started on one application process and is allowed to run on a maximum of one process. To use multiple processes, an application must be deployed with the `minimum-processes` and `maximum-processes` parameters set appropriately.

> **i Note**
>
> While an application name is assigned manually and is unique in a subaccount, an application process ID is generated automatically whenever a new process is started and is unique across the cloud platform.

Procedure

1. Open the subaccount in the cockpit.

   For information about the cockpit logon URL according to your region and host, see the *Related Information* section.
2. Choose *Applications* > *Java Applications* in the navigation area.
3. You have the following options for the applications in the list:

To...	Choose...
Start an application	 *(Start)* The application transitions to the *Started* state.
Start an additional process	 *(Start Additional Process)* The application's state continues to be shown as *Started* and an additional process appears in the *Processes* panel.
Stop an application	 *(Stop)* All running processes are stopped and the application transitions to the *Stopped* state.

To...	Choose...
Undeploy an application	 *(Delete)* The application is deleted from your subaccount and disappears from the application list. This also removes all data related to the application, such as configuration settings and logs. Data source bindings are not deleted. To delete all data source bindings created for this application, select the checkbox. **i Note** Bound databases and schemas will not be deleted. You can delete database and schema bindings using the *Databases & Schemas* panel.

4. To choose an action for an application process, click the relevant application's name in the list to go the the application overview page.
5. Choose *Monitoring* > *Processes* in the navigation area and proceed as follows:

To...	Choose...
Start an additional process	 *(Start Additional Process)* The application's state continues to be shown as *Started* and an additional process appears in the *Processes* panel.
Restart a process	 *(Restart)* The running process is stopped and a new process started. A new process ID is generated automatically.
Disable a process	 *(Disable Process)* The process state changes to *Started (disabled)*. The process continues to handle working sessions, but does not accept new connections, which allows you to shut it down gracefully.
Enable a process	 *(Enable Process)* The process state changes back to *Started*.
Stop a process	 *(Stop)* The process is stopped and removed from the list. If the application has no further processes, it transitions to the *Stopped* state.

Related Information

Regions and Hosts Available for the Neo Environment [page 17]
deploy [page 1422]
Scale Applications [page 1599]
Perform Soft Shutdown [page 1615]
Administering Database Schemas

6.4.2.2.3 Check the Process Status

The status of an individual process is based on values that reflect the process run state and its monitoring metrics.

Context

Procedure

1. In the cockpit, choose *Applications* > *Java Applications* in the navigation area.
2. Select an application in the application list by clicking the link on its name.

   This takes you to the overview page for the selected application.

   The *Processes* panel shows the number of running processes and the overall state for the metrics as follows:

   State
   - Started
   - Started (Disabled)
   - Starting
   - Stopping
   - Application Error
   - Infrastructure Error

   Metric
   - OK
   - Warning (also shown for intermediate states)
   - Critical
   - Pending
3. Choose *Monitoring* > *Processes* in the navigation area to go to the process overview to view the status summary and further details:

Panel	Description
Status Summary	Displays the current values of the two status categories and the runtime version. A short text summarizes any problems that have been detected.
State	Indicates whether the process has been started or is transitioning between the *Started* and *Stopped* states. The *Error* state indicates a fault, such as server unavailability, timeout, or VM failure.
Runtime	Shows the runtime version on which the application process is running and its current status: ○ OK: Still within the first three months since it was released ○ No longer recommended: Has exceeded the initial three-month period ○ Expired: 15 months since its release date

6.4.2.2.4 Default Trace File

Context

This page describes the format of the `Default Trace` file. You can view this file for your Web applications via the cockpit.

For more information, see Investigating Performance Issues Using the SQL Trace [page 979].

Default Trace Header

Parameter	Description
`FILE_TYPE`	The type of the `Default Trace`
`FILE_ID`	The identifier of the `Default Trace`
`ENCODING`	The encoding type used in the cloud database
`RECORD_SEPARATOR`	ASCII symbol for separating the log records. In our case, it is "**|**" (ASCII code: **124**)
`COLUMN_SEPARATOR`	ASCII symbol for separating the columns in the `Default Trace`. In our case, it is "**#**" (ASCII code: **35**)
`ESC_CHARACTER`	ASCII symbol for escape. In our case, it is "****" (ASCII code: **92**)
`COLUMNS`	Each log record contains the following information in the order displayed in the log file header: *Time*, *TZone*, *Severity*, *Logger*, *ACH*, *User*, *Thread*, *Bundle name*, *JPSpace*, *JPAppliance*, *JPComponent*, *Tenant Alias*, and *Text*.

Parameter	Description
SEVERITY_MAP	The severity map represents the following severity levels order: FINEST|Information|FINER|Information|FINE|Information|CONFIG|Information|DEBUG|Information|PATH|Information|INFO|Information|WARNING|Warning|ERROR|Error|SEVERE|Error|FATAL|Error
HEADER_END	

Besides the main log information, the `Default Trace` logs information about the tenant users that have accessed a relevant Web application. This information is provided in the new *Tenant Alias* column parameter, which is automatically logged by the runtime. The *Tenant Alias* is:

- A human-readable string;
- For new accounts, it is shorter than the tenant ID (8-30 characters);
- Unique for the relevant SAP BTP landscape;
- Equal to the account name (for new accounts); might be equal to the tenant ID (for old accounts).

Example

In this example, the application has been accessed on behalf of two tenants - with identifiers **42e00744-bf57-40b1-b3b7-04d1ca585ee3** and **5c42eee4-d5ad-494e-9afb-2be7e55d0f9c**.

```
FILE_TYPE:DAAA96DE-B0FB-4c6e-AF7B-A445F5BF9BE2
FILE_ID:1391169413918
ENCODING:[UTF8|NWCJS:ASCII]
RECORD_SEPARATOR:124
COLUMN_SEPARATOR:35
ESC_CHARACTER:92
COLUMNS:Time|TZone|Severity|Logger|ACH|User|Thread|Bundle name|JPSpace|
JPAppliance|JPComponent|Tenant Alias|Text|
SEVERITY_MAP:FINEST|Information|FINER|Information|FINE|Information|CONFIG|
Information|DEBUG|Information|PATH|Information|INFO|Information|WARNING|Warning|
ERROR|Error|SEVERE|Error|FATAL|Error
HEADER_END
2014 01 31 12:07:09#
+00#INFO#com.sap.demo.tenant.context.TenantContextServlet##anonymous#http-
bio-8041-exec-1##myaccount#myapplication#web#null#null#myaccount#The app was
accessed on behalf of tenant with ID: '42e00744-bf57-40b1-b3b7-04d1ca585ee3'|
2014 01 31 12:08:30#
+00#INFO#com.sap.demo.tenant.context.TenantContextServlet##anonymous#http-
bio-8041-exec-3##myaccount#myapplication#web#null#null#subscriberaccount#The app
was accessed on behalf of tenant with ID: '5c42eee4-d5ad-494e-9afb-2be7e55d0f9c'|
```

Related Information

Logging for the Neo Environment

Developing Multitenant Applications in the Neo Environment [page 898]

6.4.2.2.5 View Runtime Information

View information about the application runtime. SAP BTP provides a set of runtimes. You can choose the application runtime during application deployment.

Context

The runtime is assigned either by default or explicitly set when an application is deployed. If a version is not specified during deployment, the major runtime version is determined automatically based on the SDK that is used to deploy the application. By default, applications are deployed with the latest minor version of the respective major version.

You are strongly advised to use the default version, since this contains all released fixes and critical patches, including security patches. Override this behavior only in exceptional cases by explicitly setting the version, but note that this is not recommended practice.

Procedure

1. In the cockpit, choose *Java Applications* in the navigation area and then select the relevant application in the list.
   The *Runtime* panel provides the following information:
   - The application runtime name and version
   - For user-defined runtimes:
     - The major and minor versions, for example, 1.35.
     - The date until when the specified runtime version is recommended for use, or whether it is no longer recommended or has expired (also indicated by a runtime version status icon).
2. To view the actual runtime versions used by the individual processes (requires that the application is running), click the relevant application's name in the list to go the application overview page.
3. Choose *Monitoring* > *Processes* in the navigation area and select a process.
   The *Runtime* section at process level provides the following information:
   - The exact runtime version on which the process has been started (major, minor, micro, and nano versions).
   - The date until when this runtime version is recommended for use, or whether it is no longer recommended or has expired (also indicated by a runtime version status icon).

Related Information

Application Runtime Container [page 857]
Choose Application Runtime Version [page 1594]

6.4.2.3 Updating Applications

If you are an application operator and need to deploy a new version of a productive application or perform maintenance, you can choose among several approaches.

> **i** Note
>
> In all cases, first test your update in a non-productive environment. The newly deployed version of the application overwrites the old one and you cannot revert to it automatically. You have to redeploy the old version to revert the changes, if necessary.

SAP BTP provides the following approaches for updating an application:

Zero Downtime

Description: Rolling update with soft shutdown

Use: When your new application version is backward compatible with the old version - that is, the new version of the application can work in parallel with the already running old application version.

Steps: Deploy a new version of the application and disable and enable processes in a rolling manner. For an automated execution of the same procedure, use the rolling-update command.

See Update Applications with Zero Downtime [page 1610] and rolling-update [page 1534].

Planned Downtime (Maintenance Mode)

Description: Shows a custom maintenance page to end users. The application is automatically disabled.

Use: When the new version is backward incompatible - that is, running the old and the new version in parallel may lead to inconsistent data or erroneous output.

Steps: Enable maintenance mode to redirect new connections to the maintenance application. Deploy and start the new application version and then disable maintenance mode.

See Enable Maintenance Mode for Planned Downtimes [page 1613].

Soft Shutdown

Description: Supports zero downtime and planned downtime scenarios. Disabled applications/processes stop accepting new connections from users, but continue to serve already running connections.

Use: As part of the zero downtime scenario or to gracefully shut down your application during a planned downtime (without maintenance mode).

Steps: Disable the application (console client only) or individual processes (console client or cockpit) in order to shut down the application or processes gracefully.

See Perform Soft Shutdown [page 1615].

Related Information

Deploying and Updating Applications [page 875]

6.4.2.3.1 Update Applications with Zero Downtime

The platform allows you to update an application in a manner in which the application remains operable all the time and your users do not experience downtime.

Prerequisites

- You have application deployment permissions for the subaccount.
- You have at least one application process that is not in use, see your compute unit quota.
- You have downloaded and configured the SAP BTP console client. We recommend that you use the latest SDK.
  For more information, see Set Up the Console Client [page 847].

Context

Each application runs on one or more dedicated application processes. You can start one or many application processes at any given time, according to the compute unit quota that you have. Each process has a unique process ID that you can use to stop it. To update an application non-disruptively for users, you handle individual processes rather than the application as a whole. The procedure below describes the manual steps to execute a zero downtime update. Use it if you want to have more control on the respective steps, for example to have a different timeout for the different application processes before stopping them. For an automated execution of the same procedure, use the rolling-update command. For more information, see rolling-update [page 1534].

> **i Note**
>
> Not applicable to *hanatrial.ondemand.com*.

Procedure

1. Open the command prompt and navigate to the folder containing neo.bat/sh (`<SDK installation folder>/tools`).

2. List the status of the application which shows all its processes with their attributes (ID, status, last change date) by executing `<neo status>`. Identify and make a note of the application process IDs, which you will need to stop in the following steps. Application processes are listed chronologically by their last change date.

```
neo status --host <host> --account <subaccount_name> --application
<application_name> --user <e-mail_or_user>
```

3. Deploy the new version of your application on SAP BTP by executing `<neo deploy>` with the appropriate parameters.

   Note that to execute the update, you need to start one additional application process with the new version. Therefore, make sure you have configured a high enough number of maximum processes for the application (at least one higher than the number of old processes that are running). In case you have already reached the quota for your subaccount, stop one of the already running processes, before proceeding.

```
neo deploy --host <host> --account <subaccount_name> --application
<application_name> --source <file_location>
 --user <e-mail_or_user> --maximum-processes <number of maximum processes
that can be started for the application>
```

4. Start a new application process which is running the new version of the application by executing `<neo start>`.

```
neo start --host <host> --account <subaccount_name> --application
<application_name> --user <e-mail_or_user>
```

5. Use soft shutdown for the application process running the old version of the application:
   a. Execute `<neo disable>` using the ID you identified in Step 2. This command stops the creation of new connections to the application from new end users, but keeps the already running ones alive.

```
neo disable --host <host> --user <email_or_user> --application-process-id
<ID>
```

   b. Wait for some time so that all working sessions finish. You can monitor user requests and used resources by configuring JMX checks, or, you can just wait for a given time period that should be enough for most of the sessions to finish.
   c. Stop the application process by executing `<neo stop>` using the `<application-process-id>` parameter.

```
neo stop --host <host> --user <email_or_user> --application-process-id <ID>
```

6. (Optional) Make sure the application process is stopped by checking its status using the `<application-process-id>` parameter.

```
neo status --host <host> --user <email_or_user> --application-process-id <ID>
```

7. If the application is running on more than one application processes, repeat steps 4 and 5 until all the processes running the old version are stopped and the corresponding number of processes running the new version are started.

Example

For example, if your application runs on two application processes, you need to perform the following steps:

1. List the application processes running the old version:
   application process(old); application process (old)
2. Deploy the new version of the application. As you will need to start one additional application process later, make sure you have another available application process by specifying *--maximum-processes 3*. Since the newly deployed version does not start automatically, the running application processes remain unchanged:
   application process (old); application process (old)
3. Start a new application process that will use the newly deployed version:
   application process (old); application process (old); application process (new)
4. Using soft shutdown, disable and stop one of the application processes running the old version so that you have one application process with the old version and one with the new version:
   application process (new); application process (old)
5. Repeat steps 3 and 4 so that the remainng application process running the old version gets stopped and a new application process with the new application version is started instead.
   application process (new); application process (new)

Related Information

rolling-update [page 1534]
Perform Soft Shutdown [page 1615]
disable [page 1430]
deploy [page 1422]
start [page 1557]
status [page 1555]
JMX Checks [page 738]

6.4.2.3.2 Enable Maintenance Mode for Planned Downtimes

An operator can start and stop planned application downtime, during which a customized maintenance page for that application is shown to end users.

Prerequisites

To redirect an application, you need a maintenance application. A maintenance application replaces your application for a temporary period and can be as simple as a static page or have more complex logic. You need to provide the maintenance application yourself and ensure that it meets the following conditions:

- It is a Java application.
- It is deployed in the same subaccount as your application.
- It has been started, that is, it is up and running.
- It must not be in maintenance itself.
- Its context path must be the same as the context path of the original application.

Context

> **i** Note
>
> Not applicable to *hanatrial.ondemand.com*.

Cockpit

Context

You can enable the maintenance mode for an application from the overview page for the application. An application can be put into maintenance mode only if it is not being used as a maintenance application itself and is running (*Started* state).

Procedure

1. Log on to the cockpit, select a subaccount and choose *Applications* > *Java Applications* in the navigation area.
2. Click the application's name in the list to open the application overview page and in the *Application Maintenance* section choose (*Start Maintenance*).

3. In the dialog box, select the application that will serve as the maintenance application and choose *Set Selected Application*. In the application list, the application's state is now shown as *Started (In Maintenance)*.
   From this point on, new connections will be redirected to the maintenance application. All active connections will still be handled until the application is stopped.
4. Optional: To view the details in the *Application Maintenance* section, select your application in the list.
   The following details confirm that your application is in maintenance mode:
   -  *In Maintenance*
   - A link to the assigned maintenance application: Click the link to open the overview page for this application.

Results

Note that HTTP requests from already active sessions are redirected to the original application, if able. This approach makes sure that end users can complete their work without noticing the application downtime. Only new HTTP requests are redirected to the maintenance application.

The temporary redirect to the maintenance application remains effective until you take your application out of maintenance. To disable the maintenance mode, choose  (*Switch maintenance mode off*). Before doing so, you should ensure that your application is up and running to avoid end users experiencing HTTP errors.

Console Client

Procedure

1. Open the command prompt and navigate to the folder containing neo.bat/sh (`<SDK installation folder>/tools`).
2. Start the planned application downtime by executing `<neo start-maintenance>` in the command line. This stops traffic to the application and registers a maintenance page application. All active connections will be still handled until the application is stopped.

   ```
   neo start-maintenance --host <host> --account <subaccount_name> --application
   <application_name> --user <e-mail_or_user>
   --maintenance-application <maintenance application name>
   ```

   If you want to have access to an application during maintenance, use the `--direct-access-code` parameter. For more information, see start-maintenance [page 1560].
3. Perform the planned maintenance, update, or configuration of your application:
   a. Before stopping the application, wait for the working sessions to finish. You can wait for a given time period that should be enough for most of the sessions to finish, or configure JMX checks to monitor user requests and used resources. For more information, see Configure JMX Checks for Java Applications from the Console Client [page 741]

b. Stop the application by executing:

```
neo stop --host <host> --account <subaccount_name> --application
<application_name> --user <email_or_user>
```

c. Deploy the new version of your application by executing:

```
neo deploy --host <host> --account <subaccount_name> --application
<application_name> --source <file_location>
 --user <e-mail_or_user>
```

d. Start the new version of the application by executing:

```
neo start --host <host> --account <subaccount_name> --application
<application_name> --source <file_location>
 --user <e-mail_or_user>
```

4. Stop the planned application downtime by executing `<neo stop-maintenance>` in the command line. This resumes traffic to the application and the maintenance page application stops handling incoming requests.

```
neo stop-maintenance --host <host> --account <subaccount_name> --application
<application_name>  --user <e-mail_or_user>
```

Related Information

start-maintenance [page 1560]
stop-maintenance [page 1565]
deploy [page 1422]
start [page 1557]

6.4.2.3.3 Perform Soft Shutdown

Soft shutdown enables an operator to stop an application or application process in a way that no data is lost. Using soft shutdown gives sufficient time to finish serving end user requests or background jobs.

Prerequisites

You have application deployment permissions for the subaccount.

Context

Using soft shutdown, an operator can restart the application (for example, in order to update it) in a way that end users are not disturbed. First, the application process is disabled. This means that requests by users that

already have open connections to this process will be processed, but new requests will not reach this application process anymore. After the application process is disabled and remaining sessions processed, it can be stopped by the operator.

Cockpit

Context

You can disable application processes in the *Processes* panel on the application dashboard or the *State* panel on the process dashboard.

Procedure

1. Log on to the cockpit, select an subaccount and choose *Applications* > *Java Applications* in the navigation area.
2. Select an application in the application list.
3. In the *Processes* panel, choose (*Disable process*) in the relevant row. The process state changes to *Started (disabled)*.

   > **i Note**
   >
   > You can also select the process and disable it from the process dashboard.

4. Wait for some time so that all working sessions finish and then stop the process.

Related Information

Start and Stop Applications [page 1602]

Console Client

Procedure

1. Open the command prompt and navigate to the folder containing neo.bat/sh (`<SDK installation folder>/tools`).

2. Disable processing of requests from new users to the application by executing `<neo disable>` with the appropriate parameters. If you want to stop requests to a specific application process only and not to the whole application, add the `<--application-process-id>` parameter.

   ```
   neo disable --host <host> --user <e-mail_or_user> --application-process-id
   <ID>
   ```

   If you disable the entire application, or all processes of the application, then new users requesting the application will not be able to access it and will get an error.
3. Wait for some time so that all working sessions finish.

   You can monitor user requests and used resources by configuring JMX checks, or, you can just wait for a given time period that should be enough for most of the sessions to finish.
4. Stop the application by executing `<neo stop>` with the appropriate parameters. If you want to terminate a specific application process only and not the whole application, add the `<--application-process-id >`parameter.

   ```
   neo stop --host <host> --user <e-mail_or_user> --application-process-id <ID>
   ```

Related Information

disable [page 1430]
JMX Checks [page 738]

6.4.2.4 Handle Unplanned Downtime

In the event of unplanned downtime when there is no application process able to serve HTTP requests, a default error is shown to users. To prevent this, an operator can configure a custom downtime page using a downtime application, which takes over the HTTP traffic if an unplanned downtime occurs.

Prerequisites

> **i** Note
>
> Not applicable to *hanatrial.ondemand.com*.

- You have downloaded and configured the console client. We recommend that you use the latest SDK. For more information, see Set Up the Console Client [page 847].
- You have deployed and started your own downtime application in the same SAP BTP subaccount as the application itself.
- The downtime application has to be developed in a way that it returns an HTTP 503 return code. That is especially important if availability checks are configured for the original applications so that unplanned downtimes are properly detected.

Context

Procedure

1. Open the command prompt and navigate to the folder containing *neo.bat/sh* (`<SDK installation folder>/tools`).
2. Configure the downtime application by executing **`neo set-downtime-app`** in the command line.

```
neo set-downtime-app --host <host> --account <subaccount_name> --application
<application_name>  --user <e-mail_or_user>
--downtime-application <downtime_application_name>
```

3. (optional) If the downtime page is no longer needed (for example, if the original application has been undeployed), you can remove it by executing **`clear-downtime-app`** command.

```
neo clear-downtime-app --host <host> --account <subaccount_name> --
application <application_name>  --user <e-mail_or_user>
```

Related Information

set-downtime-app [page 1547]
clear-downtime-app [page 1375]

6.4.2.5 Profiling Applications

The SAP JVM Profiler helps you analyze resource-related problems in your Java application regardless of whether the JVM is running locally or on the cloud.

Typically, you first profile the application locally. Then you may continue and profile it also on the cloud. The basic procedure is the following:

1. Attach the SAP JVM Profiler to the JVM to be analyzed.
2. Analyze the retrieved profiling data in statistics and graphs.

Features

SAP JVM Profiler provides the following traces:

	**Allocation Trace**	Shows the number, size and type of the allocated objects and the methods allocating them
	**Performance Hotspot Trace**	Shows the most time-consuming methods and execution paths
	**Garbage Collection Trace**	Shows all details about the processed garbage collections
	**Method Parameter Trace**	Shows the values of a selected method parameters
	**Synchronization Trace**	Shows the most contended locks and the threads waiting for or holding them
	**File I/O Trace**	Shows the number of bytes transferred from or to files and the methods transferring them
	**Network I/O Trace**	Shows the number of bytes transferred from or to the network and the methods transferring them

Additionally, the following operations are supported:

	**Heap Dump**	Shows a complete snapshot of the Java Heap
	**Class Statistic**	Shows the classes, the number and size of their objects currently residing in the Java Heap generations

Tasks

Profile Applications Locally [page 1620]

Profile Applications on the Cloud [page 1621]

Related Information

Debugging Applications [page 895]

6.4.2.5.1 Profile Applications Locally

Overview

After you have created a Web application and verified that it is functionally correct, you may want to inspect its runtime behavior by profiling the application. This helps you to:

- Check and optimize memory usage
- Identify frequently called operations (bottlenecks and hotspots)
- Identify slow performance

> **i** Note
>
> You can use IDE of your choice for developing the application. If you want to use SAP JVM Profiler for profiling the application locally, you must use Eclipse IDE.

Prerequisites

- You have developed and deployed a Web application locally. For more information, see Deploy Locally with the Console Client [page 890].
- You have installed the SAP JVM Tools for Eclipse. For more information, see Install the SAP JVM Tools in Eclipse [page 846].
- You have installed SAP JVM. For more information, see (Optional) Install SAP JVM [page 846].

Procedure

1. Run your Web application locally. See Deploy Locally with the Console Client [page 890].
2. Go to Eclipse IDE and open the *Profiling* perspective.
3. In the *Profile* tab, choose *VM Explorer*. It will list all SAP JVM virtual machines running locally.
4. Double click the virtual machine of your local server.
5. The *Profiling* perspective is opened.
6. Choose the type of analysis you want to perform.

Result

You have successfully started a profiling run of a locally deployed Web application. You can now trigger your work load, create snapshots of the profiling data and analyze the profiling results.

Next Steps

When you have finished with your profiling session, you can stop it by disconnecting the profiling session from the *Profile* view.

Related Information

Refer to the SAP JVM Profiler documentation for details about the available analysis options. The documentation is available as part of the SAP JVM Profiler plugin in the Eclipse IDE and can be found via *Help* > *Help Contents* > *SAP JVM Profiler*.

6.4.2.5.2 Profile Applications on the Cloud

After you have created a Web application and verified that it is functionally correct, you may want to inspect its runtime behavior by profiling the application on the cloud. It is best if you first profile the Web application locally.

Prerequisites

- You have installed Eclipse IDE and the SAP JVM Tools for Eclipse. For more information, see Install the SAP JVM Tools in Eclipse [page 846].
- You have developed and deployed a Web application. For more information, see Deploying and Updating Applications [page 875].
- Optional: You have profiled your Web application locally. For more information, see Profile Applications Locally [page 1620].

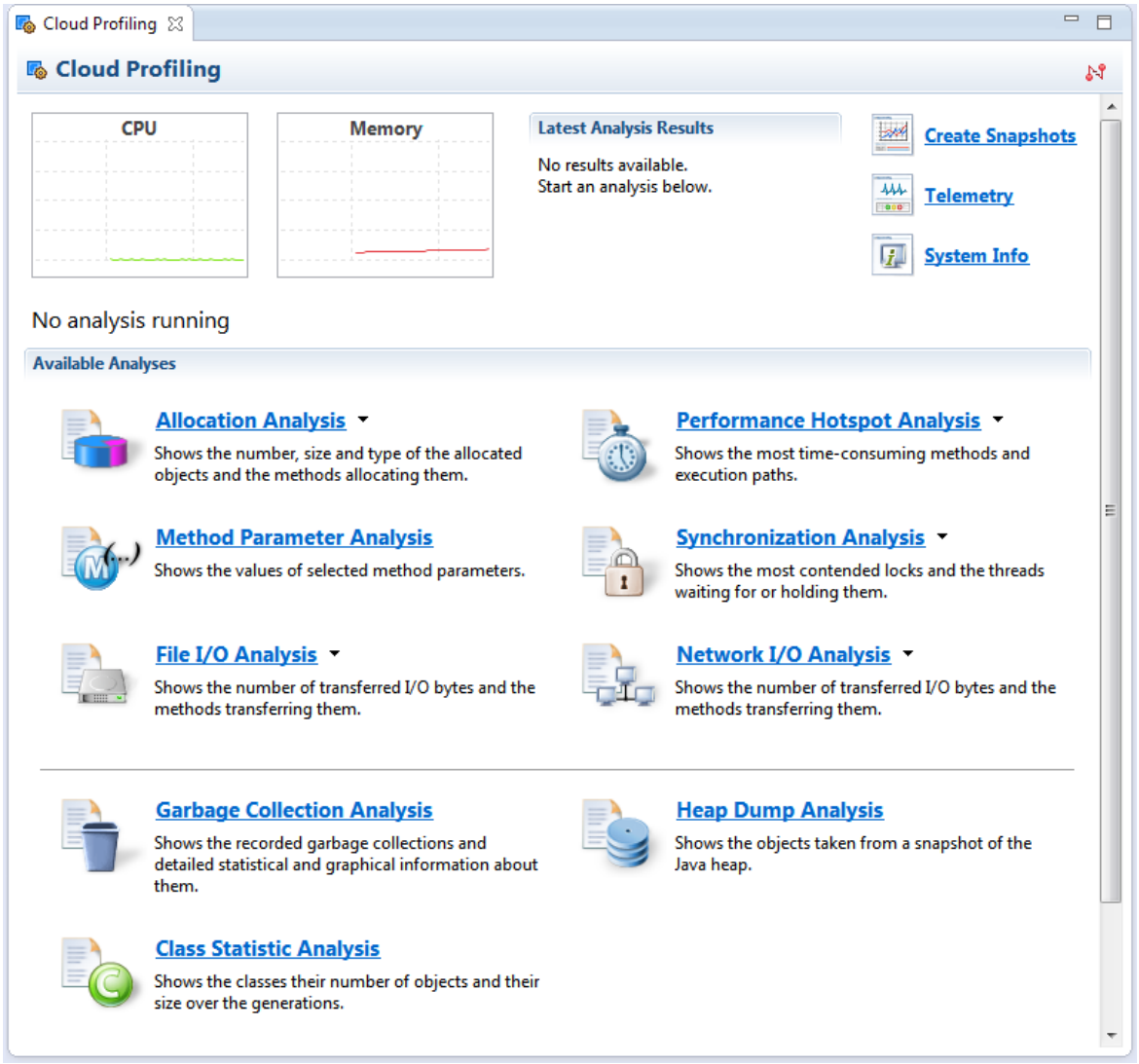
> **i Note**
>
> Currently, it is only possible to profile Web applications on the cloud that have exactly one application process (node).

Context

You can use IDE of your choice for developing the application. You must use Eclipse IDE for profiling the application on the cloud.

Procedure

1. Run your Web application on SAP BTP.
2. Open the Eclipse IDE, choose *Run*, and then *Profile Configurations*.
3. Create a new profile configuration of type *SAP HANA Cloud Profiler*.
4. In the *Project* section, choose the Java project that is already imported in the Eclipse IDE.
5. In the *Connection Properties* section, provide the application data.
6. Choose *Profile*.
7. Open the *Profiling* perspective.
8. Start a profiling analysis.



Results

You have successfully initiated a profiling run of a Web application on the cloud. Now, you can trigger your workload, create snapshots of the profiling data and analyze the profiling results.

Next Steps

When you have finished with your profiling session, you can stop it by disconnecting the profiling session from the *Profile* view.

Refer to the SAP JVM Profiler documentation for details about the available analysis options. The documentation is available as part of the SAP JVM Profiler plugin in the Eclipse IDE and you can find it via *Help* > *Help Contents* > *SAP JVM Profiler*.

6.4.2.6 Default Trace File

Context

This page describes the format of the `Default Trace` file. You can view this file for your Web applications via the cockpit.

For more information, see Investigating Performance Issues Using the SQL Trace [page 979].

Default Trace Header

Parameter	Description
`FILE_TYPE`	The type of the `Default Trace`
`FILE_ID`	The identifier of the `Default Trace`
`ENCODING`	The encoding type used in the cloud database
`RECORD_SEPARATOR`	ASCII symbol for separating the log records. In our case, it is "**|**" (ASCII code: **124**)
`COLUMN_SEPARATOR`	ASCII symbol for separating the columns in the `Default Trace`. In our case, it is "**#**" (ASCII code: **35**)
`ESC_CHARACTER`	ASCII symbol for escape. In our case, it is "****" (ASCII code: **92**)

Parameter	Description
COLUMNS	Each log record contains the following information in the order displayed in the log file header: *Time*, *TZone*, *Severity*, *Logger*, *ACH*, *User*, *Thread*, *Bundle name*, *JPSpace*, *JPAppliance*, *JPComponent*, *Tenant Alias*, and *Text*.
SEVERITY_MAP	The severity map represents the following severity levels order: FINEST|Information|FINER|Information|FINE|Information|CONFIG|Information|DEBUG|Information|PATH|Information|INFO|Information|WARNING|Warning|ERROR|Error|SEVERE|Error|FATAL|Error
HEADER_END	

Besides the main log information, the `Default Trace` logs information about the tenant users that have accessed a relevant Web application. This information is provided in the new *Tenant Alias* column parameter, which is automatically logged by the runtime. The *Tenant Alias* is:

- A human-readable string;
- For new accounts, it is shorter than the tenant ID (8-30 characters);
- Unique for the relevant SAP BTP landscape;
- Equal to the account name (for new accounts); might be equal to the tenant ID (for old accounts).

Example

In this example, the application has been accessed on behalf of two tenants - with identifiers **42e00744-bf57-40b1-b3b7-04d1ca585ee3** and **5c42eee4-d5ad-494e-9afb-2be7e55d0f9c**.

```
FILE_TYPE:DAAA96DE-B0FB-4c6e-AF7B-A445F5BF9BE2
FILE_ID:1391169413918
ENCODING:[UTF8|NWCJS:ASCII]
RECORD_SEPARATOR:124
COLUMN_SEPARATOR:35
ESC_CHARACTER:92
COLUMNS:Time|TZone|Severity|Logger|ACH|User|Thread|Bundle name|JPSpace|
JPAppliance|JPComponent|Tenant Alias|Text|
SEVERITY_MAP:FINEST|Information|FINER|Information|FINE|Information|CONFIG|
Information|DEBUG|Information|PATH|Information|INFO|Information|WARNING|Warning|
ERROR|Error|SEVERE|Error|FATAL|Error
HEADER_END
2014 01 31 12:07:09#
+00#INFO#com.sap.demo.tenant.context.TenantContextServlet##anonymous#http-
bio-8041-exec-1##myaccount#myapplication#web#null#null#myaccount#The app was
accessed on behalf of tenant with ID: '42e00744-bf57-40b1-b3b7-04d1ca585ee3'|
2014 01 31 12:08:30#
+00#INFO#com.sap.demo.tenant.context.TenantContextServlet##anonymous#http-
bio-8041-exec-3##myaccount#myapplication#web#null#null#subscriberaccount#The app
was accessed on behalf of tenant with ID: '5c42eee4-d5ad-494e-9afb-2be7e55d0f9c'|
```

Related Information

Logging for the Neo Environment
Developing Multitenant Applications in the Neo Environment [page 898]

6.4.2.7 End-to-End Trace Analysis

Trace user actions with excessive execution time within a complex system landscape using the End-to-End (E2E) trace analysis.

The End-to-End trace analysis consists of features for performing analyses throughout your entire technical landscape, so that you can isolate problematic components and identify root causes. You analyze a trace to check the distribution of the response time over the client, network, and server. As a result, the response time of each component involved in executing the request and the request path through the components are provided to you for detailed analysis.

For additional information, see Root Cause Analysis and Exception Management.

The process is covered in the following sections.

Related Information

Prerequisites for Using the End-To-End Trace Analysis [page 1625]
Develop SAP BTP Applications with E2E Tracing Analysis Support [page 1626]
Activating Statistics Gathering in SAP BTP [page 1628]
Implementing Statistics Gathering for Java Applications [page 1627]
Starting and Analyzing an End-to-End Trace [page 1629]

6.4.2.7.1 Prerequisites for Using the End-To-End Trace Analysis

Configuring Connections for Collecting Statistical Data for SAP BTP

You need to configure the connection to the SAP BTP for retrieving the statistical data. Proceed as follows, depending on the tool you use:

- Define the API service access point as described in Configuration.
- Enable the automatic trace upload as described in Tracing Capabilities.

Configuring connections for collecting statistical data for ABAP Systems

You need to configure a connection to your ABAP system for retrieving the statistical data. Proceed as follows, depending on the tool you use:

- for SAP Solution Manager 7.2 SP06 and higher - see Managing Technical System Information and Executing the Configuration Scenarios.
- for Focused Run for SAP Solution Manager (FRUN) - see Managed Systems Preparation & Maintenance Guides and Preparing Managed Systems - SAP NetWeaver Application Server ABAP.

Runtime requirements for SAP BTP

Requests passing through SAP BTP require `SAP-PASSPORT` as the HTTP header. For more information, see Approving HTTP Headers [page 1144]. When you start your tracing from an HTML5 application, then the `SAP-PASSPORT` is generated and injected automatically for you.

6.4.2.7.2 Develop SAP BTP Applications with E2E Tracing Analysis Support

HTML5 Applications

The E2E tracing is supported by default for HTML5 applications. To enable automatic upload of your business transaction started by an HTML5 application to SAP Solution Manager or FRUN, proceed as described in E2E Trace Involving SAP BTP, Neo Environment.

Java Applications

The E2E Tracing for Java applications in SAP BTP is supported. For outgoing connections to other systems, for example other Java applications in SAP BTP or on-premise systems, you must use the Connectivity Service to ensure the correct forwarding of the `SAP-PASSPORT` for all outgoing connections depending on the runtime environment.

For more information, see the corresponding sections:

Runtime	Information Location
Java EE 6 Web Profile	Preparing Managed Systems - SAP NetWeaver Consuming the Connectivity Service (Java)
Java Web Tomcat 7 Java Web Tomcat 8 Java EE 7 Web Profile TomEE 7	Implementing Statistics Gathering for Java Applications [page 1627]

6.4.2.7.2.1 Implementing Statistics Gathering for Java Applications

Context

For Java applications running on **Java Web Tomcat 7, Java Web Tomcat 8, and Java EE 7 Web Profile TomEE 7**, you have to update the `SAP-PASSPORT` and forward it as a header. To implement the tracing of outgoing connection calls, you also have to configure your destinations using the destination names from the SAP BTP cockpit, and then call them while forwarding the `SAP-PASSPORT` header.

> **i Note**
>
> All following code blocks contain example code, which might only be similar to what you have to implement in your application.

Procedure

1. Implement a custom connection in accordance with the following guidelines:
   - The implemented `ConnectionInfo` interface should return two values - a `byte[]` object id consisting of 16 bytes representing a UUID. For example, `java.util.UUID.randomUUID()` in byte array form with `UUID.mostSigBits` in bytes 0 to 7, and `UUID.leastSigBits` in Bytes 8 to 15.
   - The `ConnectionCounter int` value is a strictly-increasing value that represents the number of the request performed over the connection with the specific `ConnectionId`.
   - If you generate a new `ConnectionId` for each request the `ConnectionCounter` must always be set to `1`.
   - If you reuse a `ConnectionId` for each call, you have to increment the `ConnectionCounter` before each request.
2. Implement the acquisition of the destination URL in a way similar to the following:

   > **Sample Code**
   >
   ```
   Context ctx = new InitialContext();
   ConnectivityConfiguration configuration = (ConnectivityConfiguration)
   ctx.lookup("java:comp/env/connectivityConfiguration");
   DestinationConfiguration destConfiguration =
   configuration.getConfiguration(destinationName);
   String destinationUrl = destConfiguration.getProperty("URL");
   ```

   In the example above:
   - We instruct the look up of the connectivity configuration API
   - We set the acquisition of the destination configuration for `destinationName`.
3. Update the `SAP-PASSPORT` for the outgoing connection:

   > **Sample Code**
   >
   ```
   SapPassportHeader sapPassportHeader = updateSapPassportHeader(ctx);
   ```

```
private SapPassportHeader updateSapPassportHeader(Context ctx) throws
NamingException {
     SapPassportHeaderProvider sapPassportHeaderProvider =
(SapPassportHeaderProvider) ctx.lookup("java:comp/env/
SapPassportHeaderProvider");

        return
sapPassportHeaderProvider.getSapPassportHeader(CONNECTION_INFO);
    }
```

In the example above:

- We retrieve `SapPassportHeaderProvider` implementation from the context
- We set the current connection info. This is the connection that is established for sending/receiving data to/from a destination.

4. Set the updated `SAP-PASSPORT` as a HTTP header for the outgoing connection, and then execute the call:

Sample Code

```
HttpClient client = HttpClientBuilder.create().build();
HttpGet getRequest = new HttpGet(destionationUrl);
request.setHeader("SAP-PASSPORT",sapPassportHeader.getValue());
HttpResponse destinationResponse = client.execute(getRequest);
```

6.4.2.7.3 Activating Statistics Gathering in SAP BTP

HTML5 Applications

The E2E tracing and gathering of statistics is supported by default for HTML5 applications.

Real User Monitoring

For HTML5 applications started from the SAP Fiori Launchpad, you have to manually activate the gathering of performance statistics for each site. Proceed as follows:

1. In the SAP BTP cockpit, go to *Services* > *Portal service*.
2. In the navigation area, open the *Site Directory* tab.
3. Select the Fiori Launchpad (FLP) you want to configure and choose *Edit*. This launches the Fiori Configuration Cockpit.
4. In the navigation area, choose the *Settings* tab.
5. Choose *Edit* to change the necessary settings.
6. In the *System Settings* section, choose *On* for the *Collect Performance Data* flag.
7. Save your settings.

Java Applications

The E2E tracing and collection of data is disabled by default for Java applications. It has to be activated on demand. As prerequisites, you need a subaccount with a deployed and started Java application, you are a member of the subaccount, and you have the `Developer` role enabled.

To enable E2E tracing and collection of data, proceed as follows:

1. In the SAP BTP cockpit, navigate to your Java application.
2. In the navigation area, choose *Monitoring* > *JMX Console*.
3. Open bean *com.sap.js* > *Statistics* > *Statistics*.
4. In the *Attributes* tab, check the *Switched* property, and switch it to *false* if it is not.
5. Go to the *Operations* tab.
6. For operation `doSwitch` in parameter `p1`, enter `true`, and execute the operation by choosing the triangle icon.

You then receive an activation confirmation with the value`true` to notify you that the procedure is successful.

6.4.2.7.4 Starting and Analyzing an End-to-End Trace

Context

Procedure

To analyze an end-to-end trace, see Trace Analysis.

6.4.2.8 Using Multiple Subaccounts for Staged Application Development

SAP BTP allows you to achieve isolation between the different application life cycle stages (development, testing, productive) by using multiple subaccounts.

Prerequisites

- You have developed an application. For more information, see Developing Java Applications [page 871].
- You have a subaccount in an enterprise account. For more information, see Global Accounts.

Context

Using multiple subaccounts ensures better stability. Also, you can achieve better security for productive applications because permissions are given per subaccount.

For example, you can create three different subaccounts for one application and assign the necessary amount of compute unit quota to them:

- **dev** - use for development purposes and for testing the increments in the cloud, you can grant permissions to all application developers
- **test** - use for testing the developed application and its critical configurations to ensure quality delivery (integration testing and testing in productive-like environment prior to making it publicly available)
- **prod** - use to run productive applications, give permissions only to operators.

You can create multiple subaccounts and assign quota to them either using the console client or the cockpit.

Create Multiple Subaccounts from the Cockpit

Procedure

1. Open the cockpit and navigate to your global account.
2. Choose *New Subaccount* and specify a display name.
3. Use the "+" and "-" buttons next to quota types to set the desired amount of quota and choose the *Save* button.

   Next, you can deploy your application in the newly created subaccount using the Eclipse IDE or the console client. Then, you can test your application and make it ready for productive use.

   You can transfer the application from one subaccount to another by redeploying it in the respective subaccount.

Create Multiple Subaccounts with the Console Client

Procedure

1. Open the command prompt and navigate to the folder containing `neo.bat/sh (<SDK installation folder>/tools)`.
2. Create a new subaccount.

   Execute:

   ```
   neo create-account -a <subaccount> -u <user who is member of that subaccount> -h <host> -n <display name of the new subaccount>
   ```

3. Assign the necessary amount of quota to the subaccount.

   Execute:

   ```
   neo set-quota -a <subaccount> -u <user name or email> -h <host> -m <type of the quota lite, pro, prem or prem-plus>:<integer amount of the quota>
   ```

Next, you can deploy your application in the newly created subaccount by executing **`neo deploy -a <subaccount> -h <host> -b <application name> -s <file location> -u <user name or email>`**. Then, you can test your application and make it ready for productive use.

You can transfer the application from one subaccount to another by redeploying it in the respective subaccount.

Related Information

Org Administration Using the Cockpit
Deploying and Updating Applications [page 875]
create-account [page 1379]
delete-account [page 1400]
list-accounts [page 1478]
set-quota [page 1550]

6.4.3 SAP HANA: Application Operations

After you have developed and deployed your SAP HANA XS application, you can then monitor it.

Monitoring SAP HANA XS Applications

Cockpit	Configure Availability Checks for SAP HANA XS Applications from the Cockpit [page 1006] View Monitoring Metrics of a Database System [page 1009]
Console Client	Configure Availability Checks for SAP HANA XS Applications from the Console Client [page 1007]

6.4.4 HTML5: Application Operations

For an overview of the current status of the individual HTML5 applications in your subaccount, use the SAP BTP cockpit.

It provides key information in a summarized form and allows you to initiate actions, such as starting or stopping.

Managing Application Versions

Cockpit	Managing Application Versions [page 1635]

Managing Destinations

Cockpit	Assign Destinations for HTML5 Applications [page 1636]

Starting and Stopping Applications

Cockpit	Start and Stop HTML5 Applications [page 1637]

Managing Roles and Permissions

Cockpit	Managing Roles and Permissions [page 1637]

Monitoring

Cockpit	• View Metrics of Custom Checks for an HTML5 Application [page 754] • Configure Custom Checks for HTML5 Applications [page 755]
REST API	Metrics REST API for HTML5 Applications [page 758]

Logging

Cockpit	Using Logs in the Cockpit for HTML5 Applications [page 1642]

Related Information

Subscribe to HTML5 Multitenant Applications in the Neo Environment [page 840]

6.4.4.1 Exporting an HTML5 Application

You can export HTML5 applications either with their active version or with an inactive version.

Export the Active Version

Context

Procedure

1. Choose *Applications* > *HTML5 Applications* in the navigation area, and then the link to the application you want to export.
2. Under *Actions* in the *Overview* section, choose the export icon ( ).
3. Save the zip file.

Export an Arbitrary Version

Procedure

1. Choose *Applications* > *HTML5 Applications* in the navigation area, and then the link to the application you want to export.
2. Choose *Versioning* in the navigation area, and then choose *Versions* under *History*.
3. In the table row of the version you want to export, choose the export icon ( ).
4. Save the zip file.

6.4.4.2 Importing an HTML5 Application

You can import HTML5 applications either creating a new application or creating a new version for an existing application.

> **i** Note
>
> When you import an application or a version, the version is not imported into master branch of the repository. Therefore, the version is not visible in the history of the master branch. You have to switch to *Versions* in the navigation area.

Create a New Application

Context

Procedure

1. To upload a zip file, choose *Applications* > *HTML5 Applications* in the navigation area, and then *Import from File* (↑).
2. In the *Import from File* dialog, browse to the zip file you want to upload.
3. Enter an application name and a version name.
4. Choose *Import*.

   The new application you created by importing the zip file is displayed in the *HTML5 Applications* section.
5. To activate this version, see Activate a Version [page 1128].

Create a New Version of an Existing Application

Procedure

1. Choose *Applications* > *HTML5 Applications* in the navigation area, and then the application for which you want to create a new version.
2. Choose *Versioning* in the navigation area.
3. To upload a zip file, choose *Versions* under *History* and then *Import from File* (↑).
4. In the *Import from File* dialog, browse to the zip file you want to upload.
5. Enter a version name.
6. Choose *Import*.

The new version you created by importing the zip file is displayed in the *History* table.

7. To activate this version, select the *Activate this application version* icon ( ) in the table row for this version.
8. Confirm that you want to activate the application.

6.4.4.3 Define Application Details (HTML5 Apps)

On the *Application Details* panel, you can add or change a display name and a description to the selected HTML5 application.

Context

If a display name is maintained, this display name is also shown in the list of HTML5 applications and in the list of HTML5 subscriptions instead of the application name.

Procedure

1. Log on with a user (who is a subaccount member) to the SAP BTP cockpit.
2. Choose *Applications* > *HTML5 Applications* in the navigation area, and select the application for which to add or change a display name and description.
3. Under *Application Details* of the *Overview* section, choose *Edit*.
4. Enter a display name and a description for the HTML5 application.

Field	Comment
Display Name	Human-readable name that you can specify for your HTML5 application.
Description	Short descriptive text about the HTML5 application, typically stating what it does.

6.4.4.4 Managing Application Versions

An HTML5 application can have multiple versions, but only one of these can be active. This active version is then available to end-users of the application.

However, developers can access all versions of an application using unique URLs for testing purposes.

The *Versioning* view in the cockpit displays the list of available versions of an HTML5 application. Each version is marked either as active or inactive. You can activate an inactive version using the activation button.

For every version, the required destinations are displayed in a details table. To assign a destination from your subaccount global destinations to a required destination, choose *Edit* in the details table. By default, the

destination with the same name as the name you defined for the route in the application descriptor is assigned. If this destination does not exist, you can either create the destination or assign another one.

When you activate a version, the destinations that are currently assigned to this version are copied to the active application version.

Related Information

Create a Version [page 1127]
Activate a Version [page 1128]

6.4.4.5 Assign Destinations for HTML5 Applications

If an HTML5 application requires connectivity to one or more back-end systems, destinations must be created or assigned.

Prerequisites

A destination to the back-end system exists.

Context

For the active application version the referenced destinations are displayed in the *HTML5 Application* section of the cockpit. For a non-active application version the referenced destinations are displayed in the details table in the *Versioning* section. HTML5 applications use HTTP destinations, which can be defined on the level of your subaccount.

By default, the destination with the same name as the name you defined for the route in the application descriptor is assigned. If this destination does not exist, you can create the destination with the same name as described in Configure Destinations from the Cockpit [page 77]. Then you can assign this newly created destination. Alternatively, you can assign another destination that already exists in your subaccount. To assign a destination, follow the steps below.

Procedure

1. Log on with a user (who is a subaccount member) to the SAP BTP cockpit.
2. Choose *Applications* > *HTML5 Applications* in the navigation area, and choose the application for which you want to assign a different destination (than the default one) from your subaccount global destinations.

3. Choose *Edit* in the *Required Destinations* table.
4. In the *Mapped Subaccount Destinations* column, choose an existing destination from the dropdown list.

6.4.4.6 Start and Stop HTML5 Applications

End users can only access an application if the application is *started*. As long as an application is stopped, its end user URL does not work.

Context

The first start of the application usually occurs when you activate a version of the application.

Procedure

1. Log on with a user (who is an subaccount member) to the SAP BTP cockpit.
2. Choose *Applications* > *HTML5 Applications* in the navigation area.
3. To start your application, select it and choose the (*Start*) icon.

   The end user URL for the application is displayed under *Active Version*.
4. To stop your application, select it and choose the (*Stop*) icon.

Related Information

Navigate in the Cockpit

6.4.4.7 Managing Roles and Permissions

Resources of an HTML5 application can be protected by permissions. The application developer defines the permissions in the application descriptor file.

To grant a user the permission to access a protected resource, you can either assign a custom role or one of the predefined virtual roles to such a permission. The following predefined virtual roles are available:

- `AccountAdministrator` is equivalent to the list of subaccount members with administrator permission.
- `AccountDeveloper` is equivalent to the list of subaccount members with developer permission.
- `Everyone` is equivalent to all authenticated users of the configured Identity Provider.

`AccountDeveloper` and `AccountAdministrator` require SAP IdP to be configured as identity provider. If you want to use the `AccountDeveloper` or `AccountAdministrator` role together with a custom IDP, create those roles as custom roles and assign the corresponding user manually.

The role assignments are only effective for the active application version. To protect non-active application versions, the default permission `NonActiveApplicationPermission` is defined by the system for every HTML5 application. You can assign a role to this default permission to restrict access to the non-active version of the application.

As long as no other role is assigned to a permission, only subaccount members with developer or administrator permission have access to the protected resource. This is also true for the default permission `NonActiveApplicationPermission`.

You can create roles in the cockpit using either of these panels:

- *HTML5 Applications*: Using the HTML5 Applications Panel [page 1638]
- *Subscriptions*: Using the Subscriptions Panel [page 1640]

> **i Note**
>
> An HTML5 application's own permissions also apply when the application is reached from another HTML5 application (see Accessing Application Resources [page 1138]). Previously, only the permissions of the HTML5 application that was accessed first were considered. If you need time to assign the proper roles, you can temporarily switch back to the previous behavior by unchecking *Always Apply Permissions* in the cockpit.

Related Information

Authorization [page 1131]

6.4.4.7.1 Using the HTML5 Applications Panel

You can manage roles and permissions for the HTML5 applications or subscriptions using the *HTML5 Applications* panel.

The role management comprises the following steps:

1. Create Roles (HTML5 Applications) [page 1639]
2. Map Users or Groups to Roles (HTML5 Applications) [page 1639]
3. Assign Roles to Permissions (HTML5 Applications) [page 1640]

6.4.4.7.1.1 Create Roles (HTML5 Applications)

You create roles that are assigned to HTML5 applications or HTML5 applications subscriptions. The roles are available for all HTML5 applications and all subscriptions to HTML5 applications.

Context

Procedure

1. In the cockpit, choose *Applications* > *HTML5 Applications* in the navigation area.
2. Select any HTML5 application.
3. Choose *Roles* in the navigation area.
4. Choose *New Role*.
5. Enter the role name, and choose *Save*.

6.4.4.7.1.2 Map Users or Groups to Roles (HTML5 Applications)

You assign a role to users or a group of users.

Prerequisites

- If you want to use groups, you have configured the groups for your identity provider as described in Application Identity Provider [page 1720].

Context

Since all HTML5 applications and all HTML5 application subscriptions use the same roles, changing a role affects all applications that use this role.

Procedure

1. In the cockpit, choose *Applications* > *HTML5 Applications* in the navigation area.
2. Select any HTML5 application.
3. Choose *Roles* in the navigation area.
4. Select the role for which you want to manage assignments.
5. To assign a new user or group, choose *Assign* from the users or groups section respectively.
6. Enter the name of the user or group, and choose *Assign*.

6.4.4.7.1.3 Assign Roles to Permissions (HTML5 Applications)

Once you have created the required roles, you can assign the roles to the permissions of your HTML5 application or of your HTML5 application subscription to an HTML5 application.

Context

Procedure

1. In the cockpit, choose *Applications* > *HTML5 Applications* in the navigation area.
2. Select the HTML5 application to which you want to assign the roles.
3. The *Application Permissions* section lists all permissions defined for the selected application.
4. Choose *Edit*.
5. To assign the role to the corresponding permission, select a role from the dropdown listbox.
6. Save your entries.

6.4.4.7.2 Using the Subscriptions Panel

You can manage roles and permissions for the HTML5 applications or subscriptions using the *Subscriptions* panel.

The role management comprises the following steps:

1. Create Roles (Subscriptions) [page 1641]
2. Map Users or Groups to Roles (Subscriptions) [page 1641]
3. Assign Roles to Permissions (Subscriptions) [page 1642]

6.4.4.7.2.1 Create Roles (Subscriptions)

You create roles that are assigned to HTML5 applications or HTML5 applications subscriptions. The roles are available for all HTML5 applications and all subscriptions to HTML5 applications.

Context

Procedure

1. In the cockpit, choose *Applications* > *Subscriptions* in the navigation area.
2. From the list of subscribed HMTL5 applications, select any subscription.
3. Choose *Roles* in the navigation area.
4. Choose *New Role*.
5. Enter the role name, and choose *Save*.

6.4.4.7.2.2 Map Users or Groups to Roles (Subscriptions)

You assign a role to users or a group of users.

Prerequisites

- If you want to use groups, you have configured the groups for your identity provider as described in Application Identity Provider [page 1720].

Context

Since all HTML5 applications and all HTML5 application subscriptions use the same roles, changing a role affects all applications that use this role.

Procedure

1. In the cockpit, choose *Applications* > *Subscriptions* in the navigation area.

2. From the list of subscribed HMTL5 applications, select any subscription.
3. Choose *Roles* in the navigation area.
4. Select the role for which you want to manage assignments.
5. To assign a new user or group, choose *Assign* from the users or groups section respectively.
6. Enter the name of the user or group, and choose *Assign*.

6.4.4.7.2.3 Assign Roles to Permissions (Subscriptions)

Once you have created the required roles, you can assign the roles to the permissions of your HTML5 application or of your HTML5 application subscription to an HTML5 application.

Context

Procedure

1. In the cockpit, choose *Applications* > *Subscriptions* in the navigation area.
2. Choose the HMTL5 applications subscription to which you want to assign the roles.
3. The *Application Permissions* section lists all permissions defined for the selected application.
4. Choose *Edit*.
5. To assign the role to the corresponding permission, select a role from the dropdown listbox.
6. Save your entries.

6.4.4.8 Using Logs in the Cockpit for HTML5 Applications

You can view logs on an HTML5 application running in your subaccount or on a subscription to an HTML application.

The cockpit provides the following types of logs for an HTML5 application:

- Default traces
- HTTP access logs

You can see the logs in a log viewer, or you can download them in a log file.

> **i Note**
>
> The cockpit displays only the logs that are available for the application. For example, you see only HTTP access logs because no other logs are available for this application.

Context

Depending on the total logs size, a few files per day can be logged for each type of log (default traces and HTTP access log) . The logs are kept for 7 days before they are deleted. If the application is deleted, the logs are deleted as well. Currently, the following data is logged:

Default Traces

- The time stamp (date, time in milliseconds, time zone) of when the error occurred
- A unique request ID
- The log level (currently only ERROR and WARNING are available)
- The actual error message text

> **i** Note
>
> Each row in the log can be up to 3 KB. If the row's size is bigger than 3 KB, it's truncated.

HTML Access Logs

- The time stamp (date, time in milliseconds, time zone) of the request
- The client IP address
- The user ID
- The request method (for example GET)
- The request path suffix
- The response status code
- The request body size in bytes
- The request processing duration in milliseconds

View Logs for an HTML5 Application

Procedure

1. Open the subaccount in the SAP BTP cockpit.

   For more information, see Navigate in the Cockpit.
2. Go to the *Applications* > *HTML5 Applications* page of the subaccount.
3. Choose the relevant application to go to the overview.
   The latest logs are listed in the *Most Recent Logging* panel with the following information: type; date and time of the last modification; and size.
4. To view a more extensive list of logs, choose *Monitoring* > *Logging* in the navigation area.
   This page lists all default traces and HTTP access log files for the last week with the following information:
   - Date and time of the last modification
   - Size

   .
5. To display the contents of a particular log file, choose (*Display*). To download the file, choose (*Download*).

View Logs for a Subscription to an HTML5 Application

Procedure

1. Open the subaccount in the SAP BTP cockpit.

   For more information, see Navigate in the Cockpit.
2. Go to the *Applications* > *Subscriptions* page of the subaccount.
3. Choose the subscribed application to go to the overview.
4. Choose *Logging* in the navigation area.
   This page lists all default traces and HTTP access log files for the last week with the following information:
   - Date and time of the last modification
   - Size

   .
5. To display the contents of a particular log file, choose (*Display*). To download the file, choose (*Download*).

Related Information

Log Viewers for HTML5 Applications

6.4.5 Configuring Application URLs

By default, all applications running on SAP BTP are accessed on the *hana.ondemand.com* domain.

The URL of an application deployed in the Neo environment is `https://<application><subaccount>.<domain>`. For example, if you deploy application 'demo' in subaccount 'myshop', the application URL will be: `https://demomyshop.hana.ondemand.com`.

According to your needs, you can change the default application URL by configuring application domains different from the default one: custom or platform domains.

You can configure application domains using the console client for the Neo environment.

Note that you can use either platform domains or custom domains.

Custom Domains

Use custom domains if you want to make your applications accessible on your own domain different from *hana.ondemand.com* - for example, *www.myshop.com*. When a custom domain is used, the domain name as well as the server certificate for this domain are owned by the customer.

Platform Domains

> ⚠ Caution
>
> You can configure different platform domains only for Java applications.

By default, applications accessible on *hana.ondemand.com* are available on the Internet. Platform domains enable you to use additional features by using a platform URL different from the default one.

For example, you can use *svc.hana.ondemand.com* to hide the application from the Internet and access it only from other applications running on SAP BTP, or, *cert.hana.ondemand.com* if you want an application to use client-certificate authentication with the relevant SSL connection settings. The application URLs will be `https://demomyshop.svc.hana.ondemand.com` or, `https://demomyshop.cert.hana.ondemand.com`, respectively.

Related Information

Using Custom Domains [page 1645]
Using Platform Domains [page 1666]
Configuring Application URLs (Cloud Foundry environment)

6.4.5.1 Using Custom Domains

SAP Custom Domain service allows subaccount owners to make their SAP BTP applications accessible via a custom domain that is different from the default one (`hana.ondemand.com`) - for example **`www.myshop.com`**.

> ℹ Note
>
> If you want to configure a custom domain for a SAP Cloud Integration application, see Configuring Custom Domains for SAP Cloud Integration.

> ℹ Note
>
> Apex domains are not supported. For example, `myshop.com`.

Prerequisites

To use a custom domain for your application, you must **fulfill a number of preliminary steps**. For more information about these steps, see Prerequisites [page 1646].

Scenario

After fulfilling the prerequisites, you can configure the custom domain on your own using console client commands for the Neo environment.

First, set up secure SSL communication to ensure that your domain is trusted and all application data is protected. Then, route the traffic to your application:

1. Create an SSL Host [page 1651] - the host holds the mapping between your chosen custom domain and the application on SAP BTP as well as the SSL configuration for secure communication through this custom domain.
2. Upload a Certificate [page 1652] - it will be used as a server certificate on the SSL host.
3. Bind the Certificate to the SSL Host [page 1654].
4. Add the Custom Domain [page 1655] - this maps the custom domain to the application URL.
5. Configure DNS [page 1655]- you can create a CNAME mapping.
6. Configure Single Sign-On [page 1656] - if you have a custom trust configuration in your subaccount, you need to enable single logout.

The configuration of custom domains has different setups related to the subscriptions of your subaccount. For more information about custom domains for applications that are part of a subscription, see Custom Domains for Multitenant Applications [page 1659].

Related Information

Prerequisites [page 1646]
Configuring Custom Domains [page 1649]
Frequently Asked Questions [page 1664]
Guided Answers (Neo Environment)
No Pain, No Gain. Get Started with SAP Custom Domain Service (SAP Community Blog)
Configuring Custom Domains for SAP Cloud Integration
Configuring Custom Domains for SAP Cloud Portal service

6.4.5.1.1 Prerequisites

Before configuring an SAP custom domain, you need to perform some preliminary steps and fulfill a number of prerequisites.

> **Note**
>
> If you want to configure a custom domain for an SAP Cloud Integration application, see Configuring Custom Domains for SAP Cloud Integration.

1. Buy a custom domain quota

You need to have a quota for domains configured for your global account. One custom domain quota corresponds to one SSL host that you can use. For more information, see Purchase a Customer Account.

The following two steps involve external service providers - domain name registrar and certificate authority.

> **i Note**
>
> The domain name and the server certificate for this domain are issued by external authorities and owned by the customer.

2. Choose and buy the domain names to be used by your applications

You need to come up with a list of custom domains and applications that you want to be served through them. For example, you may decide to have three custom domains: `test.myshop.com`, `preview.myshop.com`, `www.myshop.com` - for test, preview, and productive versions of your SAP BTP application.

The domain names are owned by the customer, not by SAP BTP. Therefore, you will need to buy the custom domain names that you have chosen from a registrar selling domain names.

3. Choose an SSL certificate

To make sure that your domain is trusted and all your application data is protected, you have to get an appropriate SSL certificate from a Certificate Authority (CA).

You need to decide on the number and type of domains you want to be protected by this certificate. One SSL host can hold one SSL certificate. One certificate can be valid for a number of domains and subdomains.

There are various types of SSL certificates. Depending on your needs, you can choose between:

- **Standard certificate** - protects one domain (for example, `www.myshop.com`)
- **Wildcard certificate** - secures multiple subdomains of a domain (for example, `*.myshop.com` - this covers any subdomains of `myshop.com` - for example, `test.myshop.com`, `preview.myshop.com`, `www.myshop.com`).

  > **i Note**
  >
  > Choosing the wildcard subdomain certificate ensures protection of all subdomains in your custom domain (`*.myshop.com`), but not the domain itself (`myshop.com` cannot be used).

  Using a wildcard certificate allows you to map a large number of subdomains mapped to a single SSL host. However, this feature comes with several disadvantages:
  - If the certificate suffers from a security breach, it can affect all applications hosted on these subdomains.
  - If the HTTP traffic is too massive, it may cause performance issues for all applications hosted on these subdomains.

If there are too many custom domain mappings, consider using more SSL hosts to reduce the HTTP traffic load.

- **Subject Alternative Name (SAN) certificate** - secures multiple domain names with a single certificate. This type allows you to use any number of different domain names or common names. For example, one certificate can support: `www.myshop.com`, `*.test.myshop.com`, `*.myshop.eu`, `www.myshop.de`.

4. Generate a CSR

To issue an SSL certificate and sign it with the CA of your choice, you need a certificate signing request (CSR). You must create the CSR using our `generate-csr` command. For more information, see generate-csr [page 1446].

> ⚠ Caution
>
> The CSR is valid only for the host on which it was generated and cannot be moved and downloaded. The host represents the region: for example, *hana.ondemand.com* for Europe; *us1.hana.ondemand.com* for the United States; *ap1.hana.ondemand.com* for Asia-Pacific, and so on.

5. Issue the SSL certificate

Use the CA of your choice to sign the CSR. The certificate has to be in Privacy-enhanced Electronic Mail (PEM) format (128 or 256 bits) with private key (2048-4096 bits).

Related Information

Install the SAP BTP SDK for Neo Environment [page 845]
Set Up the Console Client [page 847]
Using the Console Client [page 1348]
Configuring Custom Domains [page 1649]
Guided Answers (Neo Environment)
Frequently Asked Questions [page 1664]

6.4.5.1.2 Configuring Custom Domains

To make sure that your domain is trusted and all application data is protected, you need to first set up secure SSL communication. The next step will then be to make your application accessible via the custom domain and route traffic to it.

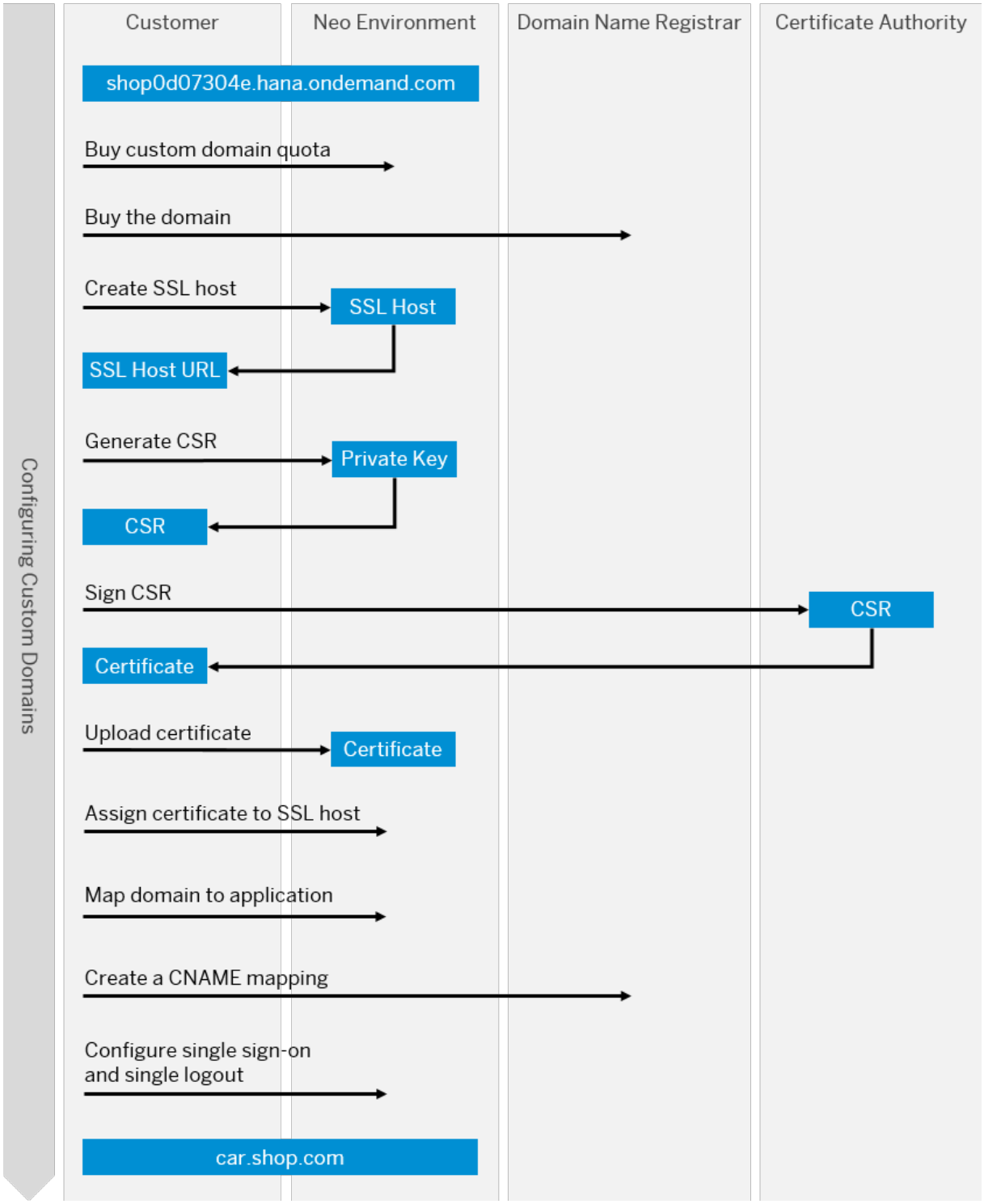
Context

This documentation is related to SAP BTP, Neo environment.

If you want to configure a custom domain:

- in the SAP BTP, Cloud Foundry environment, see Configuring Application URLs.
- for an SAP Cloud Integration application, see Configuring Custom Domains for SAP Cloud Integration.
- in the context of SAP Cloud Portal service, see Configure a Custom Domain for the Default Site.
- with regard to eDocument for Italy, see SAP Note 2583309 - eDocument Italy: Communication Channel Registration.

The end-to-end process for configuring a custom domain in the Neo environment is described in the following diagram:



For more information about purchasing the custom domain quota and the domain, see Prerequisites [page 1646].

Perform the following steps:

- Create an SSL Host [page 1651]
- Upload a Certificate [page 1652]
- Bind the Certificate to the SSL Host [page 1654]
- Add the Custom Domain [page 1655]
- Configure DNS [page 1655]
- Configure Single Sign-On and Single Logout [page 1657]
- Test the Custom Domain [page 1656]

> **i Note**
>
> For SAP Cloud Integration applications, there are some differences in the procedure. For more information, see Configuring Custom Domains for SAP Cloud Integration.

6.4.5.1.2.1 Create an SSL Host

You have to create an SSL host that will serve your custom domain. This host holds the mapping between your chosen custom domain and the application on SAP BTP as well as the SSL configuration for secure communication through this custom domain.

Prerequisites

To use the console commands, install an SDK according to the instructions in Install the SAP BTP SDK for Neo Environment [page 845].

Procedure

1. Open the command prompt and navigate to the folder containing `neo.bat/neo.sh` (`<SDK installation folder>/tools`).
2. Create an SSL host. In the console client command line, execute **`neo create-ssl-host`**. For example:

   ```
   neo create-ssl-host --account mysubaccount --user mymail@example.com --host hana.ondemand.com --name mysslhostname
   ```

   > **i Note**
   >
   > In the command output, you get the SSL host. For example, `"A new SSL host [mysslhostname] was created and is now accessible on host [123456.ssl.ondemand.com]"`. Write down the `123456.ssl.ondemand.com` host as you will later need it for the DNS configuration.

   For more information, see create-ssl-host [page 1395].

3. Optional: Check all the SSL hosts created for your subaccount using the **`list-ssl-hosts`** command.

```
neo list-ssl-hosts --account mysubaccount --user mymail@example.com --host
hana.ondemand.com
```

For more information, see list-ssl-hosts [page 1506].

6.4.5.1.2.2 Upload a Certificate

You need an SSL certificate to allow secure communication with your application. Once installed, the SSL certificate is used to identify the client/individual and to authenticate the owner of the site.

Context

The certificate generation process starts with certificate signing request (CSR) generation. A CSR is an encoded file containing your public key and specific information that identifies your company and domain name.

The next step is to use the CSR to get a server certificate signed by a certificate authority (CA) chosen by you. Before buying, carefully consider the appropriate type of SSL certificate you need. For more information, see Prerequisites [page 1646].

Procedure

1. Generate a CSR.

   The `--name` parameter is the unique identifier of the certificate within your subaccount and will be used later. It can contain alphanumeric symbols, '.', '-' and '_'.

   The `--certificate-distinguished-name` contains the attributes of the CSR formatted as type0=value0,type1=value1,type2=..., characters may be escaped by \ (backslash), no spaces are skipped. Here you need to provide the following information:

   - CN = Common Name – the domain name(s) for which you are requesting the certificate - for example ‘www.example.com’
   - C = Country - two-digit code - for example, ‘GB’
   - ST = State - state or province name - for example, ‘Hampshire’
   - L = Locality – city full name - for example ‘Portsmouth’
   - O = Organization – company name
   - OU = Organizational Unit – for example ‘IT Department’
   - E = Email Address – to validate the certificate request, some certificate authorities require the email address of the domain owner

In the console client command line, execute `neo generate-csr`:

```
neo generate-csr --account mysubaccount --user mymail@example.com --host
hana.ondemand.com
--name myfirstcert --certificate-distinguished-name
"C=GB,O=MyCompany,CN=www.example.com,E=admin@example.com"
```

For more information, see generate-csr [page 1446].

> **i Note**
>
> For security reasons, SAP recommends that you use only certificates that are based on CSRs generated via the `generate-csr` command.

In the command line output, you get the generated CSR.

2. Send the CSR to a CA. Copy and send the CSR you got in the previous step to your trusted CA who will sign the certificate.

> **i Note**
>
> When sending the CSR to be signed by a CA, make sure you choose F5 BigIP for server type.

3. Upload the SSL certificate you received from the CA:

```
neo upload-domain-certificate --account mysubaccount --user
mymail@example.com --host hana.ondemand.com --name myfirstcert --location ./
certificate.pub
```

> **i Note**
>
> The certificate must be in Privacy-enhanced Electronic Mail (PEM) format (128 or 256 bits) with private key (2048-4096 bits).

Some CAs issue chained root certificates that contain one or more intermediate certificates. In such cases, put all certificates in the file for upload starting with the signed SSL certificate.

If you did not upload an intermediate certificate for some reason, you can use the `--force` parameter option. Put the missing certificate in the file, add the `--force` parameter, and retry the previously executed `upload-domain-certificate` command without changing the values of the remaining parameters.

> **⚠ Caution**
>
> Once uploaded, the domain certificate (including the private key) is securely stored on the platform and cannot be downloaded for security reasons.

For more information, see upload-domain-certificate [page 1578].

Note that when the certificate expires, you will receive a notification from your CA. You need to take care of the certificate update. For more information, see Update an Expired Certificate [page 1660]

> **→ Tip**
>
> If you have one custom domain quota, you can upload up to four certificates (standard, wildcard, or SAN). However, you can bind only one certificate for production purposes.

> If you reach the upload limit, you can free up some space for new certificates by executing `list-domain-certificates` to get the names of the uploaded certificates and then `delete-domain-certificate` for each certificate you do not need. Deleting a certificate cannot be undone.

6.4.5.1.2.3 Bind the Certificate to the SSL Host

You need to bind the uploaded certificate to the created SSL host so that it can be used as SSL certificate for requests to this SSL host.

Procedure

1. Bind the certificate by executing **`neo bind-domain-certificate`**:

   ```
   neo bind-domain-certificate --account mysubaccount --user mymail@example.com
   --host hana.ondemand.com --ssl-host mysslhostname --certificate myfirstcert
   ```

   For more information, see bind-domain-certificate [page 1367].
2. Optional: If you want to list your custom domain certificates, execute: **`neo list-domain-certificates`**.

   ```
   neo list-domain-certificates --account mysubaccount --user mymail@example.com
   --host hana.ondemand.com
   ```

   For more information, see list-domain-certificates [page 1489].

   > **i Note**
   >
   > Optionally, you can use the `set-ssl-host` command to manage TLS protocol versions and ciphers. For more information, see set-ssl-host [page 1552].

6.4.5.1.2.4 Add the Custom Domain

To make your application on the platform accessible via the custom domain, you need to map the custom domain to the application URL.

Procedure

1. In the console client command line, execute **`neo add-custom-domain`** with the appropriate parameters. Note that you can only do this for a started application.

   ```
   neo add-custom-domain --account mysubaccount --user mymail@example.com --host
   hana.ondemand.com
   --custom-domain www.example.com --application-
   url mysubaccountmyapp.hana.ondemand.com --ssl-host mysslhostname
   ```

   > **i** Note
   >
   > Query strings are not supported in the `--application-url` parameter and are ignored. For example, if you specify "mysubaccountmyapp.hana.ondemand.com/sites?idp=example" for `--application-url`, the "?idp=example" part will be ignored.

   After you configure an application to be accessed over a custom domain, its default platform URL `hana.ondemand.com` will no longer be accessible. It will only remain accessible for subscribed applications with a URL of type `https://<application_name><provider_subaccount>-<consumer_subaccount>.<domain>`. You have the option to disable the access to the default platform URL for subscribed applications with the `--disable-application-url` parameter of the `add-custom-domain` command.

   For more information, see add-custom-domain [page 1361].
2. Optional: If you want to list all custom domains configured as access points for applications in your subaccount, execute **`neo list-custom-domain-mappings`**.

   For more information, see list-custom-domain-mappings [page 1484].

6.4.5.1.2.5 Configure DNS

To route the traffic for your custom domain to your application on the platform, you also need to configure it in the Domain Name System (DNS) that you use.

Context

You need to make a CNAME mapping from your custom domain to the created SSL host for each custom domain you want to use. This mapping is specific for the domain name provider you are using. Usually, you can modify CNAME records using the administration tools available from your domain name registrar.

Procedure

1. Sign in to the domain name registrar's administrative tool and find the place where you can update the domain DNS records.
2. Locate and update the CNAME records for your domain to point to the DNS entry you received from us (`*.ssl.ondemand.com`) - the one that you got as a result when you created the SSL host using the `create-ssl-host` command. For example, `123456.ssl.ondemand.com`. You can check the SSL host by executing the **`list-ssl-hosts`** command.

   For example, if you have two DNS records : `myhost.com` and `www.myhost.com`, you need to configure them both to point to the SSL host `123456.ssl.ondemand.com`.

   It may take some time for the configuration to take effect.

   For further details, consult your domain name registrar documentation.

6.4.5.1.2.6 Test the Custom Domain

After you configure the custom domain, make sure that the setup is correct and your application is accessible on the new domain.

Procedure

1. Log on to the cockpit, select an subaccount and go to your *Application Dashboard*. In *Application URLs*, check if the new custom URL has replaced the default one.
2. Open the new application URL in a browser. Make sure that your application responds as expected.
3. Check that there are no security warnings in the browser. View the certificate in the browser. Check the *Subject* and *Subject Alternative Name* fields - the domain names there must match the custom domain.
4. Perform a small load test - request the application from different browser sessions making at least 15 different requests.

Results

After this procedure, your application will be accessible on the custom domain, but you will not be able to log on (single sign-on) and log out (single logout), respectively. If you have a custom trust configuration in your subaccount, you will need to perform an additional configuration to enable single sign-on and single logout.

Next Steps

Configure single sign-on and single logout. For more information, see Configure Single Sign-On and Single Logout [page 1657].

6.4.5.1.2.7 Configure Single Sign-On and Single Logout

To enable single sign-on and single logout, you need to configure the *Custom Domain URLs* and the *Central Redirect URL* for the SAML single sign-on flow. To configure single sign-on and single logout, follow this procedure.

Prerequisites

- You are logged on with a user with administrator role. See Managing Member Authorizations in the Neo Environment [page 1301].
- You are aware of the productive region that hosts your subaccount. See Regions and Hosts Available for the Neo Environment [page 17].
- You are using a custom trust configuration for your subaccount. See Configure the Local Service Provider [page 1721].
- You have configured the required trust settings for your subaccount. See Configure Trust to the SAML Identity Provider [page 1724].

Context

Central Redirect URL is the central node that facilitates assertion consumer service (ACS) and single logout (SLO) service. By default, this node is provided by the platform, and has the `authn.<productive region host>.com` URL (for example, `authn.hana.ondemand.com`). If you want to use your application's root URL as the ACS, instead of the central node, you will need to maintain the *Central Redirect URL*.

For Java applications, you can follow the procedure described in the current document.

> **i Note**
>
> For HANA XS applications that use SAP ID Service as authenticating authority, create an incident in component BC-IAM-IDS. For HANA XS applications that use SAP Cloud Identity Services - Identity Authentication for authentication, see Configure a Trusted Service Provider to learn how to update the ACS and SLO endpoints.

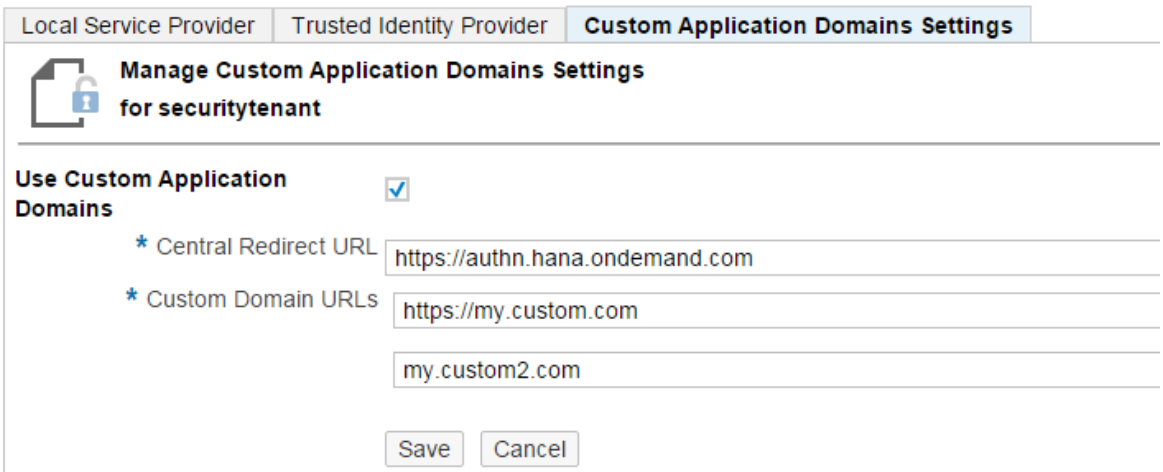
Procedure

1. In your Web browser, open the SAP BTP cockpit and choose *Security* > *Trust* in the navigation area.
2. Choose the *Custom Application Domains Settings* subtab.
3. Choose *Edit*. The custom domains properties become editable.
4. Select the *Use Custom Application Domains* option.
5. In *Central Redirect URL*, enter the URL of your application process that will serve as the central node.

> **→ Tip**
>
> The *Central Redirect URL* value has to be the same as the host of the ACS endpoint value in the metadata of the service provider.

> **i Note**
>
> Make sure you do not stop the application VM specified as the *Central Redirect URL*. Otherwise, SAML authentication will fail for all applications in your subaccount.

6. The values in *Custom Domain URLs* are used for SLO. Enter the required values (all custom domain URLs) in *Custom Domain URLs*.
7. Save your changes. The system generates the respective SLO endpoints. Test them in your Web browser and make sure they are accessible from there.



> **→ Tip**
>
> The system will accept URL values with or without `https://`. Either way, the system will generate the correct ACS and SLO endpoint URLs.

> **i Note**
>
> If you are using SAP Cloud Portal service and some of the tiles are configured to open an application on a different domain, as is the case with Web Dynpro ABAP applications, SAP recommends that you create a custom domain for each backend system. You need to add all custom domains in the *Custom Domain Application Settings* tab.

For more information about configuring a custom domain in the context of SAP Cloud Portal service, see Configure a Custom Domain for the Default Site.

6.4.5.1.3 Custom Domains for Multitenant Applications

Configuration of custom domains has different setups related to the subscriptions of your subaccount. Subscriptions represent applications that your subaccount has purchased for use from an application provider.

> **i Note**
>
> If you want to configure a custom domain for an SAP Cloud Integration application, see Configuring Custom Domains for SAP Cloud Integration.

A subscription means that there is a contract between an application provider and a tenant that authorizes the tenant to use the provider's application. As the consumer subaccount, you do not own, deploy, or operate these applications yourself. Subscriptions allow you to configure certain features of the applications and launch them through consumer-specific URLs.

The default URL of a multitenant application is: `https://<application_name><provider_subaccount>-<consumer_subaccount>.<domain>`.

When you configure custom domains for such applications that are part of a subscription, the following scenarios are possible:

- The custom domain is owned by the application provider who uses an SSL host from their subaccount quota. The provider also does the configuration and assignment of the custom domain. The provider can assign a subdomain of its own custom domain to a particular subscription URL. To do this, the provider needs to have rights in both the provider and consumer subaccount.
- The customer (consumer) uses an SSL host from the consumer subaccount quota. In this case, the customer (consumer) owns the custom domain and the SSL host and is therefore able do the necessary configuration on their own.

Related Information

Developing Multitenant Applications in the Neo Environment [page 898]

6.4.5.1.4 Managing Client Certificate Authentication for Custom Domains

If you want your customers to use client certificates when they access your application on SAP BTP via a custom domain.

Prerequisites

Your have configured a custom domain for your application. For more information, see Using Custom Domains [page 1645].

Features

As an account administrator, you can:

- Create and upload a list of trusted CA (certificate authority) certificates and assign that list to a previously created SSL host. For more information, see add-ca [page 1360].
- Configure the SSL host to optionally or mandatorily require client certificate authentication. To do that, use the `--ca-bundle` parameter when executing the `set-ssl-host` command. For more information, see set-ssl-host [page 1552].

Related Information

add-ca [page 1360]
list-cas [page 1483]
remove-ca [page 1522]
set-ssl-host [page 1552]

6.4.5.1.5 Update an Expired Certificate

When the certificate for the custom domain expires or it's about to expire, you can either upload and bind a new certificate based on a new CSR, or upload and bind a new certificate based on an already existing CSR.

Context

It's possible to update an expired certificate by uploading a signed certificate that isn't based on a CSR generated via the `generate-csr` command. However, this procedure requires you to upload a private key

from a remote location, which poses a security risk. Also, there's no way to download the uploaded private key. For more information, see upload-domain-certificate [page 1578].

Therefore, SAP recommends that you use only certificates that are based on CSRs previously generated via the `generate-csr` command.

6.4.5.1.5.1 Using a New CSR

Context

Upload and bind a new certificate to the SSL host to replace the expired certificate by generating a new CSR. If you had configured the certificate using the console client commands, follow the steps:

Procedure

1. Generate a new CSR by executing the **`neo generate-csr`** command with the appropriate parameters:

   ```
   neo generate-csr --account mysubaccount --user mymail@example.com --host
   hana.ondemand.com
   --name mynewcert --certificate-distinguished-name
   "C=BG,O=MyCompany,CN=example.com"
   ```

   For more information, see Upload a Certificate [page 1652].
2. In the command line output, you get the generated new CSR. Copy and send the text to your trusted CA to sign your certificate.
3. When you receive a signed SSL certificate from the CA, upload it by executing:

   ```
   neo upload-domain-certificate --account mysubaccount --user
   mymail@example.com --host hana.ondemand.com --name mynewcert --location ./
   certificate.pub
   ```
4. Execute **`neo set-ssl-host`** with the `--certificate` parameter.

   ```
   neo set-ssl-host --account mysubaccount --user mymail@example.com --host
   hana.ondemand.com --name mysslhostname --certificate mynewcert
   ```

   The `set-ssl-host` command allows you to unbind the expired certificate and bind the new one to the SSL host in one step. For more information, see set-ssl-host [page 1552].
5. To verify that you have configured correctly the new certificate, execute **`neo list-domain-certificates`**.

6.4.5.1.5.2 Using the CSR of the Bound Certificate

Context

Some certificate authorities (CA) offer to sign an SSL certificate based on the CSR of the already bound certificate. If you choose this option, you don't need to generate a new CSR.

> **i** Note
>
> When you update your old certificate this way, you overwrite it with the new certificate.

Procedure

1. Run the `display-csr` command to get the CSR of the certificate currently bound to your SSL host:

   ```
   neo display-csr --account mysubaccount --user mymail@example.com --host
   hana.ondemand.com --name mycurrentcert
   ```

   Then, send the CSR to your CA for signing.
2. When you receive the signed SSL certificate from your CA, upload it by executing:

   ```
   neo upload-domain-certificate --account mysubaccount --user
   mymail@example.com --host hana.ondemand.com --name mycurrentcert --location ./
   certificate.pub --force
   ```

   The `--force` parameter allows you to overwrite the old certificate and bind the new certificate in one step. For more information, see upload-domain-certificate [page 1578].

   If you don't use the `--force` parameter, you won't be able to bind the certificate to the SSL host.
3. To verify that the validity of the certificate is updated, execute **`neo list-domain-certificates`**.

Related Information

Configuring Custom Domains [page 1649]
set-ssl-host [page 1552]
upload-domain-certificate [page 1578]
bind-domain-certificate [page 1367]
unbind-domain-certificate [page 1570]
list-domain-certificates [page 1489]

6.4.5.1.6 Remove the Custom Domain

If you do not want to use the custom domain any longer, you can remove it using the console client commands. As a result, your application will be accessible only on its default hana.ondemand.com domain.

Context

Procedure

1. In the console client command line, execute **`neo remove-custom-domain`**.

   Note that you will need the SSL host name defined when configuring the custom domain. You can view it by executing **`list-ssl-hosts`**.

   ```
   neo remove-custom-domain --account mysubaccount --user mymail@example.com --
   host hana.ondemand.com
   --custom-domain www.example.com --ssl-host mysslhostname
   ```

2. Unbind the certificate.

   ```
   neo unbind-domain-certificate --account mysubaccount --user
   mymail@example.com --host hana.ondemand.com --ssl-host mysslhostname
   ```

3. Delete the certificate.

   Note that you will need the certificate name defined when configuring the custom domain.

   ```
   neo delete-domain-certificate --account mysubaccount --user
   mymail@example.com --host hana.ondemand.com --name myfirstcert
   ```

4. Delete the SSL host.

   ```
   neo delete-ssl-host --account mysubaccount --user mymail@example.com --host
   hana.ondemand.com --name mysslhostname
   ```

Related Information

remove-custom-domain [page 1523]
unbind-domain-certificate [page 1570]
delete-domain-certificate [page 1408]
delete-ssl-host [page 1412]
list-ssl-hosts [page 1506]

6.4.5.1.7 Frequently Asked Questions

Answers to some of the most commonly asked questions about SAP Custom Domain service.

How many domains (URLs) do I get to use for one custom domain?

For each custom domain that you purchase, you can create one SSL host and you can upload up to four certificates. Then, you can bind only one of these certificates to that SSL host for production purposes. The number of domains (URLs) that you can use with that single domain certificate depends on the certificate type:

- If the certificate is issued for a specific domain name (for example, `webshop.acme.com`), you can use **one domain**.
- If you use a wildcard certificate (for example, *.acme.com), the certificate is valid for **all subdomains** of `acme.com`.

  > **i Note**
  >
  > Using a wildcard certificate allows you to map a large number of subdomains mapped to a single SSL host. However, this feature comes with several disadvantages:
  > - If the certificate suffers from a security breach, it can affect all applications hosted on these subdomains.
  > - If the HTTP traffic is too massive, it may cause performance issues for all applications hosted on these subdomains.
  >
  > If there are too many custom domain mappings, consider using more SSL hosts to reduce the HTTP traffic load.

- If you use a Subject Alternative Names (SAN) certificate, you can use **many domains**. This type of certificate is usually used when multiple aliases of the same application are needed. For example, `www.acme.com`, `www.login.acme.com`.

> **i Note**
>
> Each of these options has pros and cons. It's up to you to decide which type of certificate you are going to use.

For more information, see Prerequisites [page 1646].

How many certificates can I upload?

One custom domain quota allows you to upload up to four certificates. However, you can use only one of the uploaded certificates for production purposes.

I can't configure a custom domain for my SAP Cloud Integration application. What should I do?

For SAP Cloud Integration applications, there are some differences in the procedure. When you map the custom domain to the Cloud Integration URL, keep in mind that the URL consists of several URL elements. You can find these URL elements in the cockpit. For more information, see Configuring Custom Domains for SAP Cloud Integration.

After I configure a custom domain for my application, can I still use the default hana.ondemand.com URL?

It depends. The default `hana.ondemand.com` URL remains accessible **only if the application is part of a subscription**. This type of applications has the following URL format: `https://<application_name><provider_subaccount>-<consumer_subaccount>.<domain>`. If needed, you can disable the access to the default `hana.ondemand.com` URL with the `--disable-application-url` parameter of the add-custom-domain [page 1361] command. For more information, see Custom Domains for Multitenant Applications [page 1659].

In all other cases, the default `hana.ondemand.com` becomes inaccessible and cannot be used along with the configured custom domain URL.

Is there a troubleshooting guide for custom domains?

Yes, there is. If you are facing a technical issue and you are not sure how to proceed, see Guided Answers.

Related Information

Using Custom Domains [page 1645]
Configuring Custom Domains [page 1649]

6.4.5.2 Using Platform Domains

Using platform domains, you can configure the application network availability or authentication policy. You can achieve that by configuring the appropriate platform domain which will change the URL on which your application will be accessible.

Prerequisites

You have installed and configured the console client for the Neo environment. For more information, see Using the Console Client [page 1348].

Context

The available platform domains are:

- *hana.ondemand.com* - any application is accessible on this default domain after being deployed on the platform
- *cert.hana.ondemand.com* - enables client certificate authentication
- *svc.hana.ondemand.com* - provides access within the same region; for internal communication and not open on the Internet or other networks

You can configure the platform domains using the `application-domains` group of console client commands:

- Add a Platform Domain [page 1666]
- Check Configured Domains [page 1667]
- Remove Platform Domains [page 1667]

6.4.5.2.1 Add a Platform Domain

Context

Procedure

1. Open the command prompt and navigate to the folder containing `neo.bat/neo.sh(<SDK installation folder>/tools)`.

2. Configure the platform domain you have chosen by executing the **`add-platform-domain`** command.

```
neo add-platform-domain --account mysubaccount --application myapp --user
myuser --host hana.ondemand.com --platform-domain cert.hana.ondemand.com
```

As a result, the specified application will be accessible on `cert.hana.ondemand.com` and on the default `hana.ondemand.com` domain.

6.4.5.2.2 Check Configured Domains

Context

Procedure

1. To make sure the new platform domain is configured, execute the `list-application-domains` command:

```
neo list-application-domains --account mysubaccount --application myapp --
user myuser --host hana.ondemand.com
```

2. Check if the returned list of domains contains the platform domain you set.

6.4.5.2.3 Remove Platform Domains

Context

Procedure

1. When you no longer want the application to be accessible on the configured platform domain, remove it by executing the **`remove-platform-domain`** command:

```
neo remove-platform-domain --account mysubaccount --application myapp --user
myuser --host hana.ondemand.com --platform-domain cert.hana.ondemand.com
```

2. Repeat the step for each platform domain you want to remove.

Related Information

add-platform-domain [page 1363]
list-application-domains [page 1482]
remove-platform-domain [page 1524]

6.4.6 Configuring Application Access via On-Premise Reverse Proxy

Using an on-premise reverse proxy allows you to combine on-premise and cloud-based web applications in the same browser window.

Scope

The on-premise reverse proxy method is applicable to:

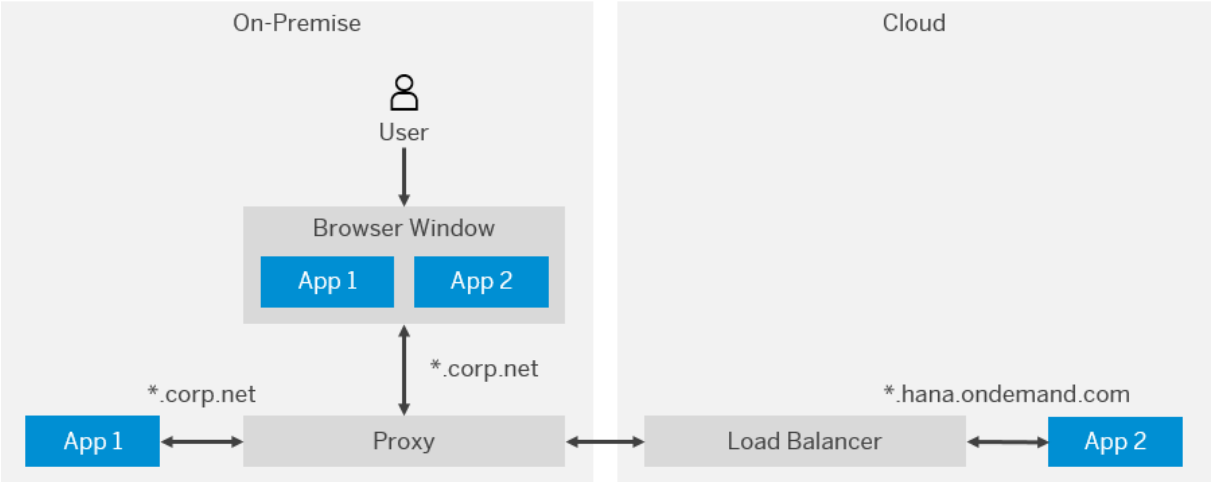
- Java applications

  > **i Note**
  >
  > A Java application must be deployed and started in advance.

- HTML5 applications
- Both host and port mapping for reverse proxy
- More than one reverse proxy address can be mapped to the same application URL.

Context

You are often not allowed to combine on-premise and cloud-based web applications in one browser window. It is not allowed because browsers use the cross-site information transfer prevention policy. Browsers count this type of information transfer as a security threat by default, which makes it impossible to perform cookie exchange and, in particular, cookie-based authentication.



You can use an on-premise reverse proxy as the sole origin for the browser. This feature allows you to manage on-premise and cloud applications in the same browser window. The commands listed in this topic allow applications running on SAP BTP to be accessed via such proxies.

> **i Note**
>
> Please have in mind that you cannot use these commands for applications configured with custom domains.

There are several options available for managing mappings between the cloud application uniform resource identifier (URI) and the proxy host. Having a proxy-to-application mapping allows access to the application via the on-premise reverse proxy.

Managing Proxy-to-Application Mappings

Open the command prompt and navigate to the folder containing `neo.bat/neo.sh` (`<SDK installation folder>/tools`). Then, you can manage the proxy host mappings by using the reverse-proxy group of console client commands:

- The `map-proxy-host` command maps an application host to an on-premise reverse proxy host and port.

  > **Example**
  >
  > ```
  > neo map-proxy-host --account mysubaccount --app-host app.hana.ondemand.com
  > --proxy loc.corp:123 -h hana.ondemand.com -u username -p ******
  > ```

- The `unmap-proxy-host` command deletes the mapping between an application host and an on-premise reverse proxy host and port.

  > **Example**
  >
  > ```
  > neo unmap-proxy-host --account mysubaccount --app-host
  > app.hana.ondemand.com --proxy loc.corp:123 -h hana.ondemand.com -u
  > username -p ******
  > ```

- The `list-proxy-host-mappings` command lists the proxy hosts mapped to an application host.

  Example

  ```
  neo list-proxy-host-mappings --account mysubaccount --app-host
  app.hana.ondemand.com --proxy loc.corp:123 -h hana.ondemand.com -u
  username -p ******
  ```

Proxy Configuration

You need to set the on-premise proxy with a header with key `x-proxy-host`. As a result, when your HTTP request arrives at the cloud, it will be routed properly to App 2. The header value should contain the application host of App 2, which is `app.hana.ondemand.com` in this specific example.

> **Note**
>
> If you do not use the `x-proxy-host` header, you will receive the `Service Unavailable` error message.

Related Information

map-proxy-host [page 1513]
unmap-proxy-host [page 1575]
list-proxy-host-mappings [page 1499]
API Documentation [page 1147]

7 Security, Neo Environment

Authorization and trust management, OAuth, key and certificate management, principal propagation and other security features in the Neo Environment.

Transport Layer Security (TLS) Connectivity Support

Neo environment uses encrypted communication channels based on HTTPS/TLS, supporting TLS version 1.2 or higher.

Make sure you use HTTP clients (such as Web browsers) that support TLS version 1.2 or higher for connecting.

In the TLS headers `x-scp-tls-version` and `x-scp-tls-cipher` returned by the Neo environment, the application receives information about the TLS version and cipher with which the connection is established.

Authorization and Trust Management in the Neo Environment [page 1672]
: The Neo environment of SAP BTP supports identity federation and single sign-on with external identity providers. The current section provides an overview of the supported scenarios.

Platform Identity Provider [page 1746]
: The platform identity provider is the user base for access to your SAP BTP subaccount in the Neo environment. The default user base is provided by SAP ID Service. You can switch to an Identity Authentication tenant if you want to use a custom user base.

OAuth 2.0 Service [page 1751]
: Use OAuth 2.0 service on SAP BTP to protect applications in the Neo environment using the OAuth 2.0 protocol.

Keystore Service [page 1779]
: The Keystore Service provides a repository for cryptographic keys and certificates to the applications in the Neo environment of SAP BTP.

Audit Logging in the Neo Environment [page 1810]
: In this section you can find information for audit log functionalities in the SAP BTP Neo environment.

Principal Propagation [page 1816]
: Exchange user ID information between systems or environments in SAP BTP.

Protection from Web Attacks [page 1835]
: To protect your applications from different kind of web attacks, Neo environment provides mechanisms for you to use with your applications.

Data Protection and Privacy [page 1848]
: Data protection is associated with numerous legal requirements and privacy concerns. In addition to compliance with general data protection and privacy acts, it is necessary to consider compliance with industry-specific legislation in different countries.

7.1 Authorization and Trust Management in the Neo Environment

The Neo environment of SAP BTP supports identity federation and single sign-on with external identity providers. The current section provides an overview of the supported scenarios.

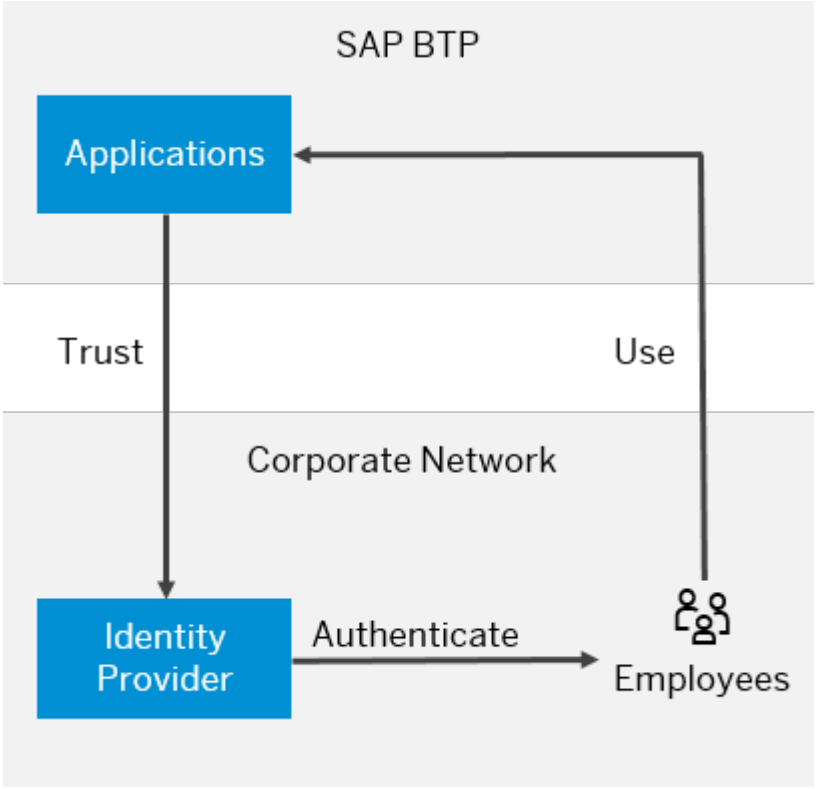
Contents

- Identity Federation with a Corporate Identity Provider [page 1672]
- Identity Federation with an Identity Authentication Tenant [page 1673]
- Default Identity Federation with SAP ID Service [page 1674]
- Managing Roles [page 1675]

Identity Federation with a Corporate Identity Provider

SAP BTP applications can delegate authentication and identity management to an existing corporate IdP that can, for example, authenticate your company's employees. It aims at providing a simple and flexible solution: your employees (or customers, partners, and so on) can single sign-on with their corporate user credentials, without a separate user store and subaccount in SAP BTP. All information required by SAP BTP about the employee can be passed securely with the logon process, based on a proven and standardized security protocol. There is no need to manage additional systems that take care for complex user account synchronization or provisioning between the corporate network and SAP BTP. Only the configuration of already existing components on both sides is needed, which simplifies administration and lowers total cost of ownership significantly. Even existing applications can be "federation-enabled" without changing a single line of code.

The following graphic illustrates this scenario.



You need to configure trust at several levels:

- At the service provider of your subaccount in SAP BTP
- Configuring trust on SAP BTP to the corporate IdP
- Configuring trust on the corporate IdP to SAP BTP

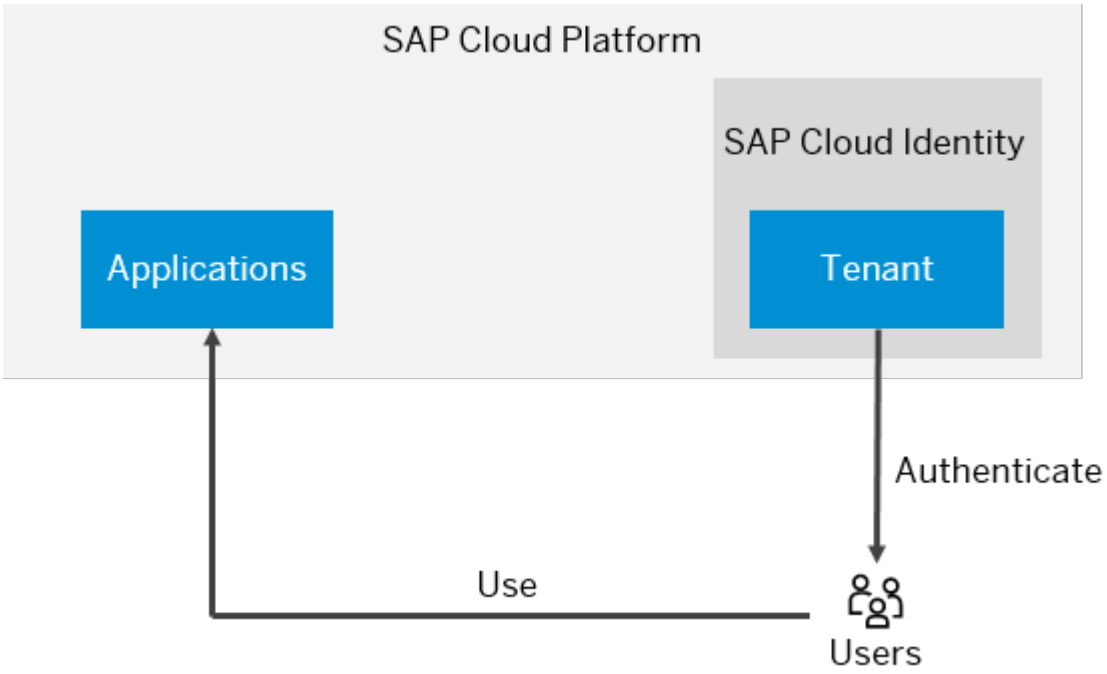
See Application Identity Provider [page 1720].

Identity Federation with an Identity Authentication Tenant

You can use Identity Authentication as an identity provider for your applications. is a cloud solution for identity lifecycle management. Using it, you can benefit from features such as user base, user provisioning, corporate branding or logo, and social IdP integration. See Identity Authentication.

Identity Authentication provides an easy way for your applications to delegate authentication and identity management and keep developers focused on the business logic. It allows authentication decisions to be removed from the application and handled in a central service.

SAP BTP offers solid integration with Identity Authentication. When you request an Identity Authentication tenant for your SAP BTP subaccount, you can automatically use it as a trusted IdP.



See Identity Authentication Tenant as an Application Identity Provider [page 1734].

Default Identity Federation with SAP ID Service

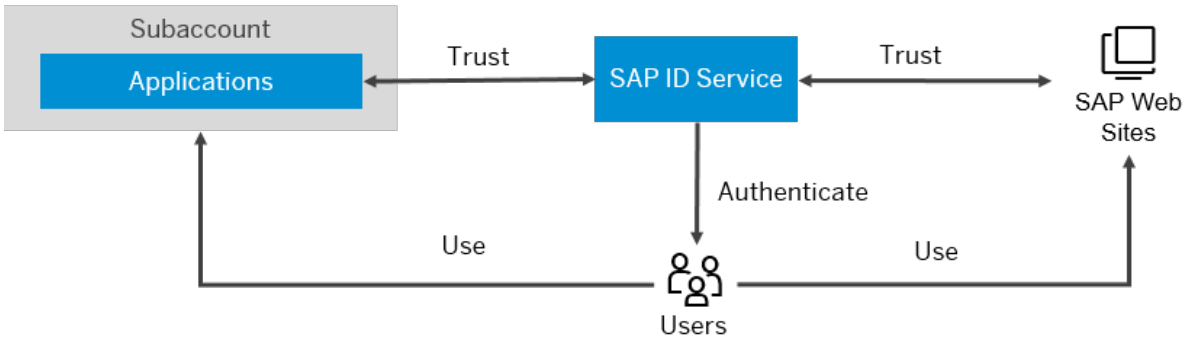
SAP ID service is the place where you have to register to get initial access to SAP BTP. If you are a new user, you can use the self-service registration option at the SAP Web site or SAP ID Service. SAP ID Service manages the users of official SAP sites, including the SAP developer and partner community. If you already have such a user, then you are already registered with SAP ID Service.

In addition, you can use SAP ID Service as an identity provider for your identity federation scenario, or if you do not want to use identity federation. Trust to SAP ID Service is pre-configured on SAP BTP by default, so you can start using it without further configuration. Optionally, on SAP BTP you can configure additional trust settings, such as service provider registration, role assignments to users and groups, and so on.

SAP ID service consists of the following main components:

- A central user store for all your identities that require access to protected resources of your application(s)
- A standards-based Single Sign-On (SSO) service that enables users to log on only once and get seamless access to all your applications deployed using SAP BTP

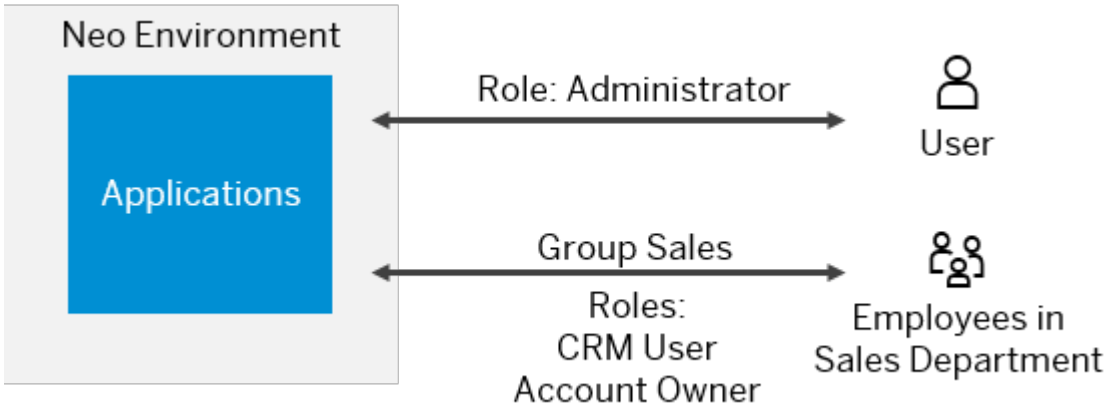
The following graphic illustrates the identity federation with SAP ID Service scenario.



Managing Roles

Roles allow you to control the access to application resources in SAP BTP, as specified in Java EE. In SAP BTP, you can assign groups or individual users to a role. Groups are collections of roles that allow the definition of business-level functions within your subaccount. They are similar to the actual business roles existing in an organization.

The following graphic illustrates a sample scenario for role, user and group management in SAP BTP. It shows a person, John Doe, with corporate role: sales representative. On SAP BTP, all sales representatives belong to group Sales, which has two roles: CRM User and Account Owner. On SAP BTP, John Doe inherits all roles of the Sales group, and has an additional role: Administrator.



See Managing Roles [page 1710].

Using an On-Premise User Store

You can use a user store from an on-premise system for user authentication scenarios. SAP BTP supports two types of on-premise user stores:

- SAP Single Sign-On
- Microsoft Active Directory

See On-Premise User Store [page 1736].

Parent topic: Security, Neo Environment [page 1671]

Related Information

Platform Identity Provider [page 1746]
OAuth 2.0 Service [page 1751]
Keystore Service [page 1779]
Audit Logging in the Neo Environment [page 1810]
Principal Propagation [page 1816]
Protection from Web Attacks [page 1835]
Data Protection and Privacy [page 1848]

7.1.1 Securing Java Applications

SAML 2.0 Authentication

SAP BTP uses the Security Assertion Markup Language (SAML) 2.0 protocol for authentication and single sign-on.

By default, SAP BTP is configured to use SAP ID service as identity provider (IdP), as specified in SAML 2.0. You can configure trust to your custom IdP, to provide access to the cloud using your own user database.

Java EE Identity and Access Management

SAP ID Service provides Identity and Access Management for Java EE Web applications hosted on SAP BTP through the mechanisms described in Java EE Servlet specification and through dedicated APIs.

Protecting Applications from Cross-Site Scripting (XSS)

Cross-site Scripting (XSS) is one of the most common types of malicious attacks to Web applications. In order to help protecting against this type of attacks, SAP BTP provides a common output encoding library to be used from applications.

Protecting from Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) is another common type of attack to Web applications. You can protect applications running on SAP BTP from CSRF, based on the Tomcat Prevention Filter.

Related Information

Application Identity Provider [page 1720]
Test Security on the Cloud (with a Local Identity Provider) [page 1703]
Managing Roles [page 1710]
Authentication [page 1678]
Protection from Cross-Site Scripting (XSS) [page 1836]
Using the Apache Tomcat CSRF Prevention Filter [page 1842]

7.1.1.1 Security Development

This section describes how you can implement security in your applications.

SAP BTP provides the following APIs for user management and authentication:

Package	Description
User Management API	

`com.sap.security.um`
`com.sap.security.um.user`
`com.sap.security.um.service`
	The user management API can be used to create and delete users or update user information.

Authentication API	
`com.sap.security.auth.login`	The authentication API provides basic login modules and callback handlers implementations and a custom LoginContext implemenatation. It relies on the user management API to provide user information required during the authentication process.
Password Storage API	
`com.sap.cloud.security.password`	The password storage API allows users to securely persist passwords and key phrases, such as passwords for keystore files.

Related Information

Authentication [page 1678]
Authorizations [page 1686]
User Attributes [page 1696]
Logout [page 1697]
OAuth 2.0 Authorization Code Grant [page 1753]
Keystore Service [page 1779]
Storing Passwords [page 1808]
Protection from Cross-Site Request Forgery [page 1840]
Protection from Cross-Site Scripting (XSS) [page 1836]

7.1.1.1.1 Authentication

In the Neo environment, enable user authentication for access to your applications.

Prerequisites

- You have installed the SAP BTP Tools for Java. See Setting Up the Development Environment [page 845].
- You have created a simple HelloWorld application. See .
- If you want to use Java EE 6 Web Profile features in your application, you have downloaded the SAP BTP SDK for Java EE 6 Web Profile. See

Context

- Declarative Authentication [page 1679]
- Programmatic Authentication [page 1683]
- Handling Session Timeout [page 1684]
- Troubleshooting [page 1685]

> **i Note**
>
> User names in SAP BTP are case insensitive.

7.1.1.1.1.1 Declarative Authentication

Context

The Java EE servlet specification allows the security mechanisms for an application to be declared in the `web.xml` deployment descriptor.

SAP BTP supports the following default authentication methods:

Authentication Method	Default Options	Description	Sample Usecase
FORM	Trusted SAML 2.0 identity provider Application-to-Application SSO	FORM authentication implemented over the Security Assertion Markup Language (SAML) 2.0 protocol. Authentication is delegated to SAP ID service or custom identity provider. You can specify the custom identity provider using the trust configuration for your subaccount. See Application Identity Provider [page 1720]. (Optional) If you configure a connection with an on-premise user store, the existence of the user is also verified in the on-premise SAP NetWeaver AS Java system. See Using an SAP System as an On-Premise User Store [page 1737].	You want to delegate authentication to your corporate identity provider.

Authentication Method	Default Options	Description	Sample Usecase
BASIC	User name and password	HTTP BASIC authentication delegated to SAP ID service or an on-premise SAP NetWeaver AS Java system. Web browsers prompt users to enter a user name and password. By default, SAP ID service is used. (Optional) If you configure a connection with an on-premise user store, the authentication is delegated to an on-premise SAP NetWeaver AS Java system. See Using an SAP System as an On-Premise User Store [page 1737]. **ℹ Note** If you want to use your Identity Authentication tenant for BASIC authentication (instead of SAP ID service/SAP NetWeaver), create a customer ticket in component **BC-NEO-SEC-IAM**. In the ticket, specify the technical name of the subaccount, region and Identity Authentication tenant you want to use. **! Restriction** BASIC authentication with a third-party corporate identity provider is **not** supported. **! Restriction** The trust configuration (*cloud cockpit* > *Security* > *Trust* > *Application Identity Provider*) you set for your subaccount does	Example 1: You want to delegate authentication to SAP ID service. Users will log in with their SCN user name and password. Example 2: You have an on-premise SAP NetWeaver AS Java system used as a user store. You want users to log in using the user name and password stored in AS Java.

Authentication Method	Default Options	Description	Sample Usecase
		not apply for BASIC authentication.	
CERT	Client certificate	Used for authentication only with client certificate. See Enabling Client Certificate Authentication [page 1797].	Users log in using their corporate client certificates.
BASICCERT	User name and password Client certificate	Used for authentication either with client certificate or with user name and password. See Enabling Client Certificate Authentication [page 1797].	Within the corporate network, users log in using their client certificates. Outside that network, users log in using user name and password.
OAUTH	OAuth 2.0 token	Authentication according to the OAuth 2.0 protocol with an OAuth access token. See OAuth 2.0 Authorization Code Grant [page 1753]	You have a mobile application consuming REST APIs using the OAuth 2.0 protocol. Users log in using an OAuth access token.
SAML2	Trusted SAML 2.0 identity provider Application-to-Application SSO	See FORM.	See FORM.

If you need to configure the default options of an authentication method, or define new methods, see Authentication Configuration [page 1742]

→ Tip

We recommend using FORM or OAUTH authentication method.

ℹ Note

By default, any other method (DIGEST, CLIENT-CERT, etc. or custom) that you specify in the *web.xml* are executed as FORM. You can configure those methods using the *Authentication Configuration* section at Java application level in the Cockpit. See Authentication Configuration [page 1742].

→ Tip

For the SAML and FORM authentication methods, if your application sends multiple simultaneous requests without an authenticated session, they may fail. We recommend that you first send one request to a protected resource, establish a session, and then use the session for the multiple simultaneous requests.

→ Tip

Although BASIC authentication is usually used for technical users to consume REST services (stateless communication) we recommend that the client leverages the security session instead of sending credentials with every call. This means the client needs to make sure it preserves and re-sends all HTTP cookies it receives. Thus, authentication will happen only once and this could improve performance.

Results

- When FORM authentication is used, you are redirected to SAP ID service or another identity provider, where you are authenticated with your user name and password. The servlet content is then displayed.
- When BASIC authentication is used, you see a popup window and are prompted to enter your credentials. The servlet content is then displayed.

Example

Example 1: Using FORM Authentication

The following example illustrates using FORM authentication. It requires all users to authenticate before accessing the protected resource. It does not, however, manage authorizations according to the user roles - it authorizes all authenticated users.

```
<login-config>
  <auth-method>FORM</auth-method>
</login-config>
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Protected Area</web-resource-name>
    <url-pattern>/index.jsp</url-pattern>
    <url-pattern>/a2asso.jsp</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <!-- Role Everyone will not be assignable -->
    <role-name>Everyone</role-name>
    </auth-constraint>
</security-constraint>
<security-role>
  <description>All SAP BTP users</description>
  <role-name>Everyone</role-name>
</security-role>
```

> **i Note**
>
> All authenticated users implicitly have the Everyone role. You cannot remove or edit this role. In the SAP BTP Cockipt, the Everyone role is not listed in role mapping (see Managing Roles [page 1710]).

Example 2: Using FORM Authentication with Roles

If you want to manage authorizations according to user roles, you should define the corresponding constraints in the *web.xml*. The following example defines a resource available for users with role Developer, and another resource for users with role Manager:

```
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Developer Page</web-resource-name>
      <url-pattern>/developer.jsp</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>Developer</role-name>
    </auth-constraint>
  </security-constraint>
  <security-constraint>
    <web-resource-collection>
```

```
      <web-resource-name>Manager Page</web-resource-name>
      <url-pattern>/manager.jsp</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>Manager</role-name>
    </auth-constraint>
  </security-constraint>
  <login-config>
    <auth-method>FORM</auth-method>
  </login-config>
```

→ Remember

If you define roles in the *web.xml*, you need to manage the role assignments of users after you deploy your application on SAP BTP. See Managing Roles [page 1710]

7.1.1.1.1.2 Programmatic Authentication

Context

With programmatic authentication, you do not need to declare constrained resources in the *web.xml* file of your application. Instead, you declare the resources as public, and you decide in the application logic when to trigger authentication. In this case, you have to invoke the authentication API explicitly before executing any application code that should be protected. You also need to check whether the user is already authenticated, and should not trigger authentication if the user is logged on, except for certain scenarios where explicit re-authentication is required.

If you trigger authentication in an SAP BTP application protected with FORM, the user is redirected to SAP ID service or custom identity provider for authentication, and is then returned to the original application that triggered authentication.

If you trigger authentication in an SAP BTP application protected with BASIC, the Web browser displays a popup window to the user, prompting him or her to provide a user name and password.

```
package hello;
import java.io.IOException;
import javax.security.auth.login.LoginContext;
import javax.security.auth.login.LoginException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.sap.security.auth.login.LoginContextFactory;
public class HelloWorldServlet extends HttpServlet {

...
  protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
    String user = request.getRemoteUser();
    if (user != null) {
      response.getWriter().println("Hello, " + user);
    } else {
      LoginContext loginContext;
      try {
        loginContext = LoginContextFactory.createLoginContext("FORM");
        loginContext.login();
```

```
        response.getWriter().println("Hello, " + request.getRemoteUser());
      } catch (LoginException e) {
        e.printStackTrace();
      }
    }
    ...
  }

  protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
    doGet(request, response);
  }
  ...
}
```

In the example above, you create *LoginContext* and call its `login()` method.

> **i Note**
>
> All the steps below are described using the FORM authentication method, but they can also be applied to BASIC.

Procedure

1. Open the source code of your HelloWorldServlet class. Add the code for programmatic authentication to the `doGet()` method.
2. Make the `doPost()` method invoke programmatic authentication. This is necessary because the SAP ID service always returns the SAML2 response over an HTTP POST binding, and in order to be processed correctly, the *LoginContext* login must be called during the `doPost()` method. The authentication framework is responsible for restoring the original request using GET after successful authentication. Another alternative is that your `doPost()` method simply calls your `doGet()` method.
3. Test your application on the local server. It does not need to be connected to the SAP ID service, and authentication is done against local users. For more information, see Testing User Authentication on the Local Server.
4. Deploy the application to SAP BTP. If you are using FORM, you are redirected to SAP ID service or another identity provider, depending on your trust configuration for this subaccount. If you are using BASIC, you are redirected to SAP ID service (**not** configurable using trust settings). The servlet content is then displayed and you should be able to see the content returned by the hello servlet.

   When BASIC authentication is used, you should see a popup window prompting you to provide credentials to authenticate. Once these are entered successfully, the servlet content is displayed.

7.1.1.1.1.3 Handling Session Timeout

You can configure session timeout using the *web.xml*. Default value: 20 minutes. For example:

```
<session-config>
    <session-timeout>15</session-timeout> <!-- in minutes -->
  </session-config>
```

After the specified timeout, user sessions are invalidated. If the user tries to access an invalidated session, SAP BTP will return a login page in its response, so the user can enter credentials again. . If you are using SAML as login protocol, you cannot rely on the response code to find out the your session is expired because it will be 200 or 302. To check whether the response is for triggering new login, get the `com.sap.cloud.security.login` HTTP header, and reload the page. For example:

```
jQuery(document).ajaxComplete(function(e, jqXHR){
                if(jqXHR.getResponseHeader("com.sap.cloud.security.login")){
                                                alert("Session is expired, page
shall be reloaded.");
                                                window.location.reload();
                }
}
```

> **Note**
>
> For requests made with the `X-Requested-With` header and value `XMLHttpRequest` (AJAX requests), you need to check for session expiration (by checking the marker header `com.sap.cloud.security.login`). If the session is expired and you are using SAML2 or FORM authentication method, the system does not trigger an authentication request.

7.1.1.1.1.4 Troubleshooting

Use the SAP Community, SAP Support Portal and Guided Answers or other related tools as described in Getting Support, Neo Environment [page 1859].

Support Components

Use the following components if you need to create a ticket for Authorization and Trust Management in the Neo environment:

Suport Component	Description
BC-NEO-SEC-IAM	Main support channel for tickets concerning Authorization and Trust Management in the Neo environment.
BC-NEO-SEC-CPG	Support for cryptographic services, including client certificate authentication and Keystore service.

Guided Answers

Use the Security in the Neo Environment guided answers to locate the relevant solutions to problems or answers to questions.

Local Testing

When testing in the local scenario, and your application has `Web-ContextPath: /`, you might experience the following problem with Microsoft Internet Explorer:

After authentication you see:

> **Output Code**
>
> ```
> HTTP Status 405 - HTTP method POST is not supported by this URL
> ```

If you see such issues, you will have to add the following code into your protected resource:

```
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException { doGet(req, resp); }
```

Next Steps

You can now test the application locally. See Test Security Locally [page 1701].

After testing, you can proceed with deploying the application to SAP BTP. See Deploying and Updating Applications [page 875].

After deploying on SAP BTP, you need to configure the role assignments users and groups will have for this application. See Managing Roles [page 1710].

Optionally, you can configure the authentication options applied in the authentication method that you defined in the *web.xml* or programmatically. See Authentication Configuration [page 1742].

Example

To see the end-to-end scenario of managing roles on SAP BTP, watch the complete video tutorial Managing Roles in SAP BTP .

7.1.1.1.2 Authorizations

Use the `isUserInRole` method of the servlet request (`javax.servlet.HttpServletRequest`) to manage authorizations for users. The following example returns a 403 error if the user accessing this servlet does not have role Developer:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    PrintWriter out = response.getWriter();

    if(!request.isUserInRole("Developer")){
      response.sendError(403, "Logged in user does not have role Developer");
      return;
    } else {
      out.println("Hello developer");
    }
  }
```

Next Steps

You can now test the application locally. For more information, see Test Security Locally [page 1701].

After testing, you can proceed with deploying the application to SAP BTP. For more information, see Deploying and Updating Applications [page 875].

After deploying on SAP BTP, you need to configure the role assignments users and groups will have for this application. For more information, see Managing Roles [page 1710].

7.1.1.1.3 Authorization Management API

The Authorization Management API allows you to manage user roles and groups, and their assignments in your applications.

The Authorization Management API is protected with *OAuth 2.0 client credentials*. Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149].

> **i Note**
>
> We strongly recommend that you use this API only for administration, not for runtime checks of authorizations. For the runtime checks, we recommend using `HttpServletRequest.isUserInRole(java.lang.String role)`. See Authorizations [page 1686].

> **i Note**
>
> HTML5 applications are using a more feature-rich authorization model, which allows to assign *permissions* on various *URI paths*. Those permissions are then mapped to SAP BTP *custom roles*. Since all HTML5 applications are run via a central app called *dispatcher* from the *services* account – all of them share the same custom roles and mappings. This the reason why when you are managing roles of HTML5 applications, in the API calls you need to use *dispatcher* for *appName* and *services* for *providerAccount* name.

Example: Accessing Roles

1. At application level, create an HTTP destination with the following information:
   - Name=<your destination name>
   - URL=https://api.<cloud platform host>/authorization/v1
   - ProxyType=Internet
   - Type=HTTP
   - Authentication=NoAuthentication
   - CloudConnectorVersion=2

   See Create HTTP Destinations [page 80].
2. In your application, obtain an `HttpURLConnection` object that uses the destination.
   See ConnectivityConfiguration API [page 139].

3. With the object retrieved from the previous step, execute a GET call.

```
//value is the URL property of the destination
URL url = new URL(value + "/accounts/{account}/applications/{application}/
roles");

HttpURLConnection urlConnection = (HttpURLConnection)url.openConnection();
urlConnection.setRequestMethod("GET");
urlConnection.setRequestProperty("Authorization", "Bearer <token ID from
previous step>");
urlConnection.connect();
```

A response returning the roles could be:

```
{
    "roles": [
        {
            "name": "Developer",
            "type": "PREDEFINED",
            "applicationRole": true,
            "shared": true
        },
        {
            "name": "Administrator",
            "type": "PREDEFINED",
            "applicationRole": true,
            "shared": true
        }
    ]
}
```

Related Information

Authorization Management API
Using Platform APIs [page 1149]

7.1.1.1.4 Platform Authorization Management API

The Platform Authorization Management API allows you to manage the users authorized to access your subaccount in the Neo environment.

Overview

The Platform Authorization Management API is implemented over the System for Cross-domain Identity Management (SCIM) protocol. The HTTP requests that you send to this API need to be SCIM-compliant.

The Platform Authorization Management API is protected with *OAuth 2.0 client credentials*. Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149]. The required scopes for the token are: `readAccountMembers` and `manageAccountMembers`.

API URL:

```
https://api.<cloud platform host>/authorization/v1/platform/accounts/
<subaccount>/Users
```

For the cloud platform host, see Regions and Hosts Available for the Neo Environment [page 17].

ServiceProviderConfig Endpoint:

```
https://api.<cloud platform host>/authorization/v1/platform/ServiceProviderConfig
```

By the SCIM specification, you can access this endpoint (using HTTP GET) to retrieve the API configuration and its supported features. This endpoint is unprotected.

Filtering Users

You can do two types of filtering: based on the user ID or the user base. For more information about changing and managing the user base, see Platform Identity Provider [page 1746].

> ! Restriction
>
> Only the **eq** operator is supported for filtering. For more information, see Section 3.4.2.2: Filtering

Use the following URL:

```
https://api.<cloud platform host>/authorization/v1/platform/accounts/
<subaccount>/Users?filter=username eq "<user name>"
```

or

```
https://api.<cloud platform host>/authorization/v1/platform/accounts/
<subaccount>/Users?filter=userbase eq "<user base>"
```

> i Note
>
> The above two URLs are case insensitive. For more information, see Section 7.8: Case-Insensitive Comparison and International Languages of the SCIM protocol specification.

The Platform Authorization Management API returns two types of user roles: `Predefined` and `Custom`. For more information about roles, see Managing Roles [page 1710].

If a user comes from a custom user base (that is, your custom Identity Authentication tenant, not SAP ID service), the Platform Authorization Management API returns the user ID with a suffix `_<your Identity Authentication tenant>`.

Example 1: Getting the Attributes of a User Assigned to This Subaccount

A prerequisite is having a valid OAuth access token with the required scopes. See Using Platform APIs [page 1149].

1. At application level, create an HTTP destination with the following information:
   - Name=<your destination name>

- URL=https://api.<cloud platform host>/authorization/v1/platform/accounts/<subaccount>/Users/<user ID>
- ProxyType=Internet
- Type=HTTP
- Authentication=NoAuthentication
- CloudConnectorVersion=2

See Create HTTP Destinations [page 80].

2. In your application, obtain an `HttpURLConnection` object that uses the destination.
   See ConnectivityConfiguration API [page 139].
3. With the object retrieved from the previous step, execute a GET call.

```
//value is the URL property of the destination
URL url = new URL(value + "/accounts/<my subaccount>/Users/<userID>");

HttpURLConnection urlConnection = (HttpURLConnection)url.openConnection();
urlConnection.setRequestMethod("GET");
urlConnection.setRequestProperty("Authorization", "Bearer <OAuth token ID>");
urlConnection.connect();
```

A response returning the list of users could be:

```
{
            "id": "P1234567",
            "meta": {
                "created": "2019-01-16T18:01:57.105Z",
                "lastModified": "2019-01-16T18:01:57.105Z",
                "location": "https://api.hana.ondemand.com/authorization/v1/
platform/accounts/myaccount/Users/P1234567"
            },
            "schemas": [
                "com:sap:cloud:security:platform:1.0:UserExt",
                "urn:ietf:params:scim:schemas:core:2.0:User"
            ],
            "userName": "P1234567",
            "name": {
                "familyName": "Smith",
                "givenName": "John"
            },
            "emails": [
                {
                    "value": "jsmith@mycompany.com",
                    "primary": true
                }
            ],
            "roles": [
                {
                    "value": "AccountAdministrator",
                    "primary": false
                },
                {
                    "value": "Developer",
                    "primary": false
                }
            ],
            "com:sap:cloud:security:platform:1.0:UserExt": {
                "userbase": "accounts.sap.com"
            }
}
```

Example 2: Filtering Users by User Name

A prerequisite is having a valid OAuth access token with the required scopes. See Using Platform APIs [page 1149]

As with the previous example, you can use an HTTP destination object for convenience in managing your connections. For testing purposes, you can access directly the API using an HTTP GET request. Let's try to get all users with name P1234567. Then our HTTP URL looks like that:

```
https://api.<cloud platform host>/authorization/v1/platform/accounts/
<subaccount>/Users?filter=username eq "P1234567"
```

A response returning the list of users with the same name available in different user bases could be:

```
{
"Resources": [
    {
        "id": "P1234567",
        "meta": {
            "created": "2019-02-12T13:54:14.604Z",
            "lastModified": "2019-02-12T13:54:14.604Z",
            "location": "https://api.hana.ondemand.com/authorization/v1/platform/
accounts/<subaccount>/Users/P1234567"
        },
        "schemas": [
            "urn:sap:cloud:scim:schemas:extension:custom:2.0:UserExt",
            "urn:ietf:params:scim:schemas:core:2.0:User"
        ],
        "userName": "P1234567",
        "roles": [
            {
                "value": "AccountAdministrator",
                "primary": false,
                "type": "Predefined"
            },
            {
                "value": "Developer",
                "primary": false,
                "type": "Predefined"
            }
        ],
        "urn:sap:cloud:scim:schemas:extension:custom:2.0:UserExt": {
            "userbase": "ACCOUNTS.SAP.COM"
        }
    },
    {
        "id": "P1234567_ACCOUNTS400.SAP.COM",
        "meta": {
            "created": "2019-02-12T13:54:14.604Z",
            "lastModified": "2019-02-12T13:54:14.604Z",
            "location": "https://api.hana.ondemand.com/authorization/v1/platform/
accounts/<subaccount>/Users/P1234567_ACCOUNTS400.SAP.COM"
        },
        "schemas": [
            "urn:sap:cloud:scim:schemas:extension:custom:2.0:UserExt",
            "urn:ietf:params:scim:schemas:core:2.0:User"
        ],
        "userName": "P1234567",
        "roles": [
            {
                "value": "AccountAdministrator",
                "primary": false,
                "type": "Predefined"
            },
```

```
            {
                "value": "Developer",
                "primary": false,
                "type": "Predefined"
            }
        ],
        "urn:sap:cloud:scim:schemas:extension:custom:2.0:UserExt": {
            "userbase": "ACCOUNTS400.SAP.COM"
        }
    },
    {
        "id": "P1234567_ACCOUNTS500.SAP.COM",
        "meta": {
            "created": "2019-02-12T13:54:14.604Z",
            "lastModified": "2019-02-12T13:54:14.604Z",
            "location": "https://api.hana.ondemand.com/authorization/v1/platform/
accounts/<subaccount>/Users/P1234567_ACCOUNTS500.SAP.COM"
        },
        "schemas": [
            "urn:sap:cloud:scim:schemas:extension:custom:2.0:UserExt",
            "urn:ietf:params:scim:schemas:core:2.0:User"
        ],
        "userName": "P1234567",
        "roles": [
            {
                "value": "AccountAdministrator",
                "primary": false,
                "type": "Predefined"
            },
            {
                "value": "Developer",
                "primary": false,
                "type": "Predefined"
            }
        ],
        "urn:sap:cloud:scim:schemas:extension:custom:2.0:UserExt": {
            "userbase": "ACCOUNTS500.SAP.COM"
        }
    }
],
"totalResults": 3,
"itemsPerPage": 3,
"startIndex": 1,
"schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
]
}
```


Related Information

Platform Authorization Management API

Using Platform APIs [page 1149]

7.1.1.1.5 Using Platform APIs

Platform APIs are protected with *OAuth 2.0 client credentials*. Create an OAuth client and obtain an access token to call the platform API methods.

Context

For description of OAuth 2.0 client credentials grant, see the OAuth 2.0 client credentials grant specification .

For some of the platform APIs, the JWT type of access tokens is used. See Best Practices for Resilient OAuth 2.0 Communication [page 1768].

For detailed description of the available methods, see the respective API documentation.

> → Tip
>
> Do not get a new OAuth access token for each and every platform API call. Re-use the same existing access token throughout its validity period instead, until you get a response indicating the access token needs to be re-issued.

1. Create an OAuth Client

Context

The OAuth client is identified by a *client ID* and protected with a *client secret*. In a later step, those are used to obtain the OAuth API access token from the *OAuth access token endpoint*.

Procedure

1. In your Web browser, open the SAP BTP cockpit.
2. Go to the *Security* > *OAuth* section.
3. Enter the *Platform API* tab.
4. Choose *Create API Client*.
5. Choose the required API option and required combination of scopes. We recommend that you enter an OAuth client description as well.
6. Save the client.
7. Copy the generated client ID and client secret, and save them locally.

   > ⚠ Caution
   >
   > Make sure you save the generated client credentials. Once you close the confirmation dialog, you cannot retrieve the generated client credentials again.

2. Get an OAuth Access Token

Context

OAuth access token endpoint and use the *client ID* and *client secret* as user and password for HTTP Basic Authentication. You will receive the access token as a response.

By default, the access token received in this way is valid 1500 seconds (25 minutes). You cannot configure its validity length.

If you want to revoke the access token before its validity ends, delete the respective OAuth client. The access token remains valid up to 2 minutes after the client is deleted.

Procedure

1. Send a POST request to the OAuth access token endpoint. The URL is landscape specific, and looks like this:

   https://api.<cloud platform host>/oauth2/apitoken/v1?grant_type=client_credentials

   See Regions.

   The parameter *grant_type=client_credentials* notifies the endpoint that the Client Credentials flow is used.
2. Get and save the access token from the received response from the endpoint.

   The response is a JSON object, whose *access_token* parameter is the access token. It is valid for the specified time (in seconds) in the *expires_in* parameter. (default value: 1500 seconds).

Example

Retrieving an access token on the trial landscape will look like this:

```
POST https://api.hanatrial.ondemand.com/oauth2/apitoken/v1?
grant_type=client_credentials
Headers:
Authorization: Basic eW91ckNsaWVudElEOnlvdXJDbGllbnRTZWNyZXQ
```

The `eW91ckNsaWVudElEOnlvdXJDbGllbnRTZWNyZXQ` String in the above request is the Base-64 encoded `<clientID>:<ClientSecret>`.

You receive a response like this:

```
{
    "access_token": "51ddd94b15ec85b4d54315b5546abf93",
    "token_type": "Bearer",
    "expires_in": 1500,
    "scope": "hcp.manageAuthorizationSettings hcp.readAuthorizationSettings"
}
```

Alternatively, you can do this using an HTTP destination.

1. At the required (application, subaccount or global account) level, create an HTTP destination with the following information (the name can be different):
   - Name=<yourdestination name>
   - URL=https://api.<cloud platform host>/oauth2/apitoken/v1?grant_type=client_credentials
   - ProxyType=Internet
   - Type=HTTP
   - CloudConnectorVersion=2
   - Authentication=BasicAuthentication
   - User=<clientID>
   - Password=<clientSecret>

   See Create HTTP Destinations [page 80].
2. In your application, obtain an `HttpURLConnection` object that uses the destination.
   See ConnectivityConfiguration API [page 139].
3. With the object retrieved from the previous step, execute a POST call.

```
urlConnection.setRequestMethod("POST");
urlConnection.setRequestProperty("Authorization", "Basic <Base64 encoded
representation of {clientId}:{clientSecret}>");
urlConnection.connect();
```

3. Call API Methods

Procedure

In the requests to the required platform API, include the access token as a header with name *Authorization* and value *Bearer <token value>*.

Example

We will call the Authorization Management API.

```
GET https://api.hanatrial.ondemand.com/authorization/v1/accounts/p1234567trial/
users/roles/?userId=myUser
Headers:
Authorization: Bearer 51ddd94b15ec85b4d54315b5546abf93
```

Related Information

Keystore API [page 1779]
Authorization Management API [page 1687]
Audit Log Retrieval API Usage for the Neo Environment [page 1810]
Audit Log Retention API Usage for the Neo Environment [page 1815]

Platform Authorization Management API [page 1688]

7.1.1.1.6 User Attributes

You can access user attributes using the User Management Java API (`com.sap.security.um.user`). It can be used to get and create users or to read and update their information.

To access user information, you need to get to `com.sap.security.um.user.UserProvider`.

To get `UserProvider`, first, declare a resource reference in the *web.xml*. For example:

```
<resource-ref>
    <res-ref-name>user/Provider</res-ref-name>
    <res-type>com.sap.security.um.user.UserProvider</res-type>
</resource-ref>
```

Then look up `UserProvider` via JNDI in the source code of your application. For example:

```
InitialContext ctx = new InitialContext();
UserProvider userProvider = (UserProvider) ctx.lookup("java:comp/env/user/
Provider");
User user = null;
if (request.getUserPrincipal() != null) {
     user = userProvider.getUser(request.getUserPrincipal().getName());
}
```

> **i Note**
>
> If you are using the SDK for Java EE 6 Web Profile, you can look up `UserProvider` via annotation (instead of embedding JNDI lookup in the code). For example:
>
> ```
> @Resource
> private UserProvider userProvider;
> try {
>   // Read the currently logged in user from the user storage
>   return userProvider.getUser(request.getRemoteUser());
> } catch (PersistenceException e) {
>   throw new ServletException(e);
> }
> ```

Alternatively, you can access `UserProvider` using `com.sap.security.um.user.UserManagementAccessor`. For example:

```
import com.sap.security.um.user.User;
import com.sap.security.um.user.UserProvider;
import com.sap.security.um.service.UserManagementAccessor;
...
// Check for a logged in user
if (request.getUserPrincipal() != null) {
  try {
    // UserProvider provides access to the user storage
    UserProvider users = UserManagementAccessor.getUserProvider();
    // Read the currently logged in user from the user storage
    User user = users.getUser(request.getUserPrincipal().getName());
    // Print the user name and email
    response.getWriter().println("User name: " + user.getAttribute("firstname")
+ " " + user.getAttribute("lastname"));
```

```
    response.getWriter().println("Email: " + user.getAttribute("email"));

    response.getWriter().println("Assigned groups: " +
Arrays.toString(user.getAttributeValues("groups")));

  } catch (Exception e) {
    // Handle errors
  }
}
```

In the source code above, the `user.getAttribute` method is used for single-value attributes (the first name and last name of the user). For attributes that we expect to have more than one value (such as the assigned groups), we use `user.getAttributeValues` method.

Next Steps

You can now test the application locally. For more information, see Test Security Locally [page 1701].

After testing, you can proceed with deploying the application to SAP BTP. For more information, see Deploying and Updating Applications [page 875].

7.1.1.1.7 Logout

This topic describes how to enable users to log out from your applications.

Context

You can provide a logout operation for your application by adding a logout button or logout link.

When logout is triggered in a SAP BTP application, the user is redirected to the identity provider to be logged out there, and is then returned to the original application URL that triggered the logout request.

The following code provides a sample servlet that handles logout operations. When `loginContext.logout()` is used, the system automatically redirects the logout request to the identity provider, and then returns the user to the logout servlet again.

```
import javax.security.auth.login.LoginContext;
import javax.security.auth.login.LoginException;
import com.sap.security.auth.login.LoginContextFactory;
...
public class LogoutServlet extends HttpServlet {
. . .
  //Call logout if the user is logged in
  LoginContext loginContext = null;
  if (request.getRemoteUser() != null) {
    try {
      loginContext = LoginContextFactory.createLoginContext();
      loginContext.logout();

    } catch (LoginException e) {
```

```
      // Servlet container handles the login exception
      // It throws it to the application for its information
      response.getWriter().println("Logout failed. Reason: " + e.getMessage());
    }
   } else {
    response.getWriter().println("You have successfully logged out.");
   }
. . .
}
```

We add a logout link to the HelloWorld servlet, which references this logout servlet:

```
      response.getWriter().println("<a href=\"LogoutServlet\">Logout</a>");
```

(Optional) Protecting Logout from Cross-Site Request Forgery (CSRF)

CSRF is a common Web hacking attack. For more information, see Cross-Site Request Forgery (CSRF) (non-SAP link). You might consider protecting the logout operations for your applications from CSRF to prevent your users from potential CSRF attack related problems (for example, XSRF denial of service on single logout).

> **i Note**
>
> Although SAP BTP provides ready-to-use support for CSRF filtering, with logout operations you cannot use it. The reason is users are sent to the logout servlet twice: first, when they trigger logout by clicking a button/link, and second, when the identity provider has logged them out and redirected them back to the application. You cannot specify the system to apply the CSRF filter first time, and skip it the second time.

The following example provides XSRF-protected logout.

> **Source Code**

```
LoginContext loginContext = null;
    if (request.getRemoteUser() != null) {
      Object csrfTokenFromSession = request.getSession().getAttribute("csrf-
logout");
      String csrfTokenFromRequest = request.getParameter("csrf-logout");
      if (request.getSession(false) != null && csrfTokenFromRequest != null
&& csrfTokenFromSession != null
          && csrfTokenFromSession.toString().equals(csrfTokenFromRequest)) {
        try {
          loginContext = LoginContextFactory.createLoginContext();
          loginContext.logout();
        } catch (LoginException e) {
          // Servlet container handles the login exception
          // It throws it to the application for its information
          response.getWriter().println("Logout failed. Reason: " +
e.getMessage());
        }
      } else {
        response.sendError(403, "No valid csrf token found in request. No
logout will be performed.");
      }
    } else {
      response.getWriter().println("You have successfully logged out.");
    }
```

We add a logout link to the HelloWorld servlet, which references this logout servlet:

Source Code

```
try {
          HttpSession session = request.getSession(false);
          if(session != null){
            long tokenValue = 0L;
            if(session.getAttribute("csrf-logout") != null){
              tokenValue = (Long) session.getAttribute("csrf-logout");
            } else {
              SecureRandom instance =
java.security.SecureRandom.getInstance("SHA1PRNG");
              instance.setSeed(instance.generateSeed(5));
              tokenValue = instance.nextLong();
              session.setAttribute("csrf-logout", tokenValue);
            }
         response.getWriter().println("<a href=\"LogoutServlet?csrf-
logout="+tokenValue+"\">Logout</a>");
       }
     } catch (NoSuchAlgorithmException e) {
       throw new ServletException(e);
     }
```

Preventing Common Logout Problems

Unprotect the Logout Servlet

For efficient logout to work, the servlet handling logout must not be protected in the *web.xml*. Otherwise, requesting logout will result in a login request. The following example illustrates how to unprotect successfully a logout servlet. The additional *<security-constraint>...</security-constraint>* section explicitly enables access to the logout servlet.

```
<security-constraint>
    <web-resource-collection>
      <web-resource-name>Start Page</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
<auth-constraint>
            <role-name>Everyone</role-name>
</ auth-constraint>
  </security-constraint>
<security-constraint>
  <web-resource-collection>
   <web-resource-name>Logout</web-resource-name>
   <url-pattern>/LogoutServlet</url-pattern>
  </web-resource-collection>
 </security-constraint>
```

Avoid Mapping Servlet Resources to / in the web.xml*

Avoid mapping a servlet to resources using wildcard (*<url-pattern>/*</url-pattern>* in the *web.xml*). This may lead to an infinite loop. Instead, map the servlet to particular resources, as in the following example:

```
  <servlet-mapping>
    <servlet-name>Logout Servlet</servlet-name>
    <url-pattern>/LogoutServlet</url-pattern>
    <servlet-class>test.LogoutServlet</servlet-class>
```

```
</servlet-mapping>
```

Next Steps

You can now test the application locally. For more information, see Test Security Locally [page 1701].

After testing, you can proceed with deploying the application to SAP BTP. For more information, see Deploying and Updating Applications [page 875].

7.1.1.1.8 Commmon Errors with Basic Authentication in SAP ID Service

This section describes the error messages you may encounter when using BASIC authentication with SAP ID Service as an identity provider.

For more information about using BASIC authentication, see Authentication [page 1678].

Error Messages

Error Message	Description
Your account is temporarily locked. It will be automatically unlocked in 60 minutes.	SAP ID Service has registered five unsuccessful login attempts for this account in a short time. For security reasons, your account is disabled for 60 minutes.
Password authentication is disabled for your account. Log in with a certificate.	The owner of this account has disabled password authentication using their user profile settings in SAP ID service.
Inactive account. Activate it via your account creation confirmation email	This is a new account and you haven't activated it yet. You will receive an e-mail confirming your account creating, and containing an account activation link.
Login failed. Contact your administrator.	You cannot log in for a reason different from all others listed here.

7.1.1.2 Security Testing

This section describes how you can test the security you have implemented in your Java applications.

First, you need to test your application on your local runtime. If you use the Eclipse Tools, you can easily test with local users. This is useful if you are implementing role-based identity management in your application.

Then, if everything goes well on the local runtime, you can deploy your application on SAP BTP, and test how the application works on the Cloud with your local SAML 2.0 identity provider. This makes use if you are implementing SAML 2.0 identity federation.

Related Information

Test Security Locally [page 1701]
Test Security on the Cloud (with a Local Identity Provider) [page 1703]

7.1.1.2.1 Test Security Locally

When you add user authentication to your application, you can test it first on the local server before uploading it to SAP BTP.

> **i Note**
>
> On the local server, authentication is handled locally, that is, not by the SAP ID service. When you try to access a protected resource on the local server, you will see a local login page (not SAP ID service's or another identity provider's login page). User access is then either granted or denied based on a local JSON (JavaScript Object Notation) file (*<local_server_dir>/config_master/com.sap.security.um.provider.neo.local/neousers.json*), which defines the local set of user accounts, along with their roles and attributes. This is just for testing purposes. When you deploy to the cloud, user authentication is still handled by the SAP ID service.

Test Locally with the Console Client

To test security locally, you need to find and edit manually the JSON file configuring local test users. It is located at *<local_server_dir>/config_master/com.sap.security.um.provider.neo.local/neousers.json*.

The following example shows a sample configuration of a JSON file with two users, along with their attributes and roles: first user with id `john` and password `johndoe`, and second user `jane` with password `janedoe`.

```
{
  "Users": [
    {
      "UID": "john",
      "Password": "{SSHA}Ub53I5XdyH/Nh3gr3pUQ6vPyjYPPRVKF",
      "Roles": [
        "Everyone"
      ],
      "Attributes": [
        {
          "attributeName": "firstname",
          "attributeValue": "John"
        },
        {
          "attributeName": "lastname",
          "attributeValue": "Doe"
        },
        {
          "attributeName": "email",
          "attributeValue": "john.doe@sap.com"
        }
      ]
    },
```

```
    {
      "UID": "jane",
      "Password": "{SSHA}/obq802EmKc+rc4/fjU/XzbvBiaHtNIH",
      "Roles": [
        "Employee",
               "Manager"
      ],
      "Attributes": [
        {
          "attributeName": "firstname",
          "attributeValue": "Jane"
        },
        {
          "attributeName": "lastname",
          "attributeValue": "Doe"
        },
        {
          "attributeName": "email",
          "attributeValue": "jane.doe@sap.com"
        }
      ]
    }
  ]
}
```

> **Note**
>
> `SSHA` is a password storage scheme. When developing locally, you can use the `slappasswd` utility – an `OpenLDAP` password utility for Linux, Mac OS X, or other Unix-based OS – to generate such hashes:
>
> ```
> slappasswd -h {SSHA} -s <password>
> ```

Troubleshooting

When stopping your local server, you might see the following error logs:

```
#ERROR#org.apache.catalina.core.ContainerBase##anonymous#System Bundle
Shutdown###ContainerBase.removeChild: stop:
org.apache.catalina.LifecycleException: Failed to stop component
[StandardEngine[Catalina].StandardHost[localhost].StandardContext[/idelogin]]
```

This error causes *no* harm and you don't need to take any measures.

Next Steps

- After testing, you can proceed with deploying the application to SAP BTP. For more information, see Deploying and Updating Applications [page 875].
- After deploying on the cloud, you may need to perform configuration steps using the cockpit. For more information, see Security Configuration [page 1710].

7.1.1.2.2 Test Security on the Cloud (with a Local Identity Provider)

You can use a local test identity provider (IdP) to test single sign on (SSO) and identity federation of an SAP BTP application end-to-end.

This scenario offers simplified testing in which developers establish trust to an application deployed in the cloud with an easy-to-use local test identity provider .

For more information about the identity provider concept in SAP BTP, see Application Identity Provider [page 1720].

Contents:

- Prerequisites [page 1703]
- Procedure [page 1703]
  - 1. Set up the local test IdP [page 1703]
  - 2. Configure the service provider of your account in SAP BTP [page 1704]
  - 3. (Optional) Configure the local IdP name [page 1705]
  - 4. Configure trust on SAP BTP to the local test IdP [page 1705]
  - 5. Generate self sign-key pair and certificate for the local test IdP (optional) [page 1708]
  - 6. Configure trust on the local test IdP to SAP BTP [page 1708]

Prerequisites

- You have set up and configured the Eclipse IDE for Java EE Developers and SAP BTP Tools for Java. For more information, see Setting Up the Tools and SDK [page 845].
- You have developed and deployed your application on SAP BTP. For more information, see *Creating an SAP BTP Application*.

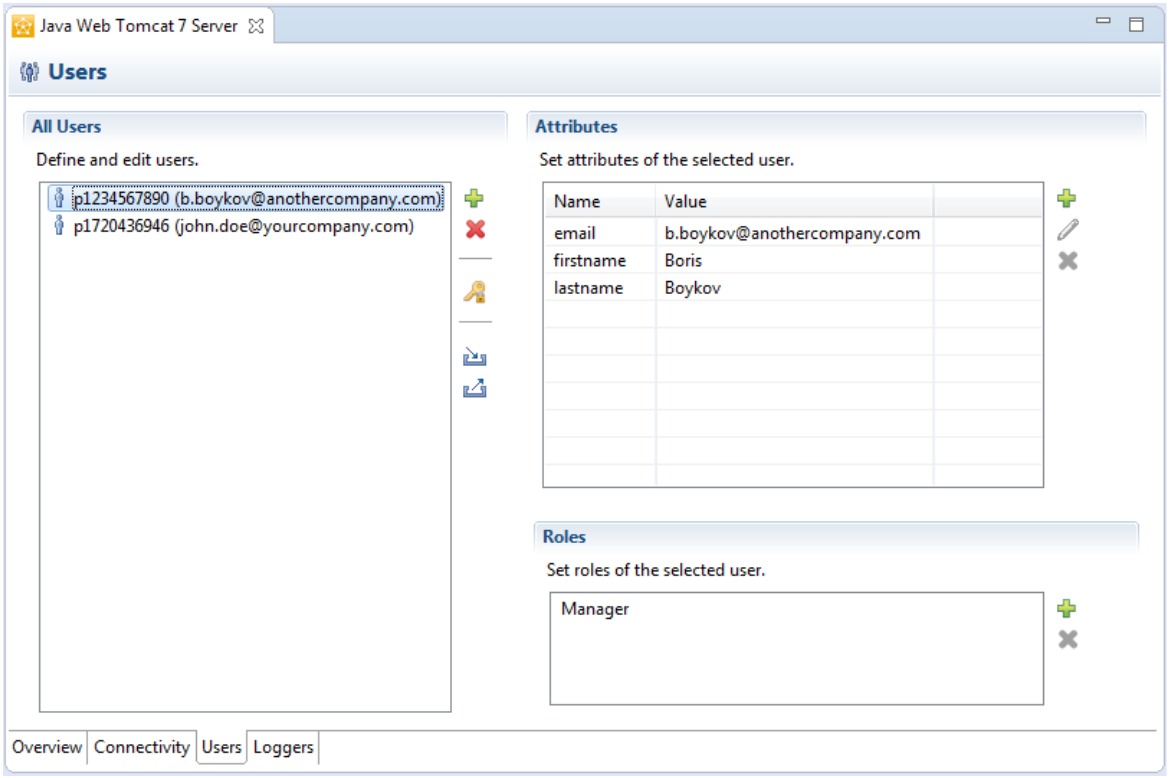
Procedure

The usage of the local test identity provider involves the following steps:

1. Set up the local test IdP

1. Open the Eclipse IDE.
2. Go to the *Servers* view.
3. From the context menu, choose *New* > *Server* .
4. In the *Define a new server* wizard, select *Java Web Server*, *Java Web Tomcat 7 Server*, or *Java EE 6 Web Profile Server* (depending on the SDK you use).
5. Start the server. The local test IdP is packaged within the SDK, so when you start the server, it will start as well.

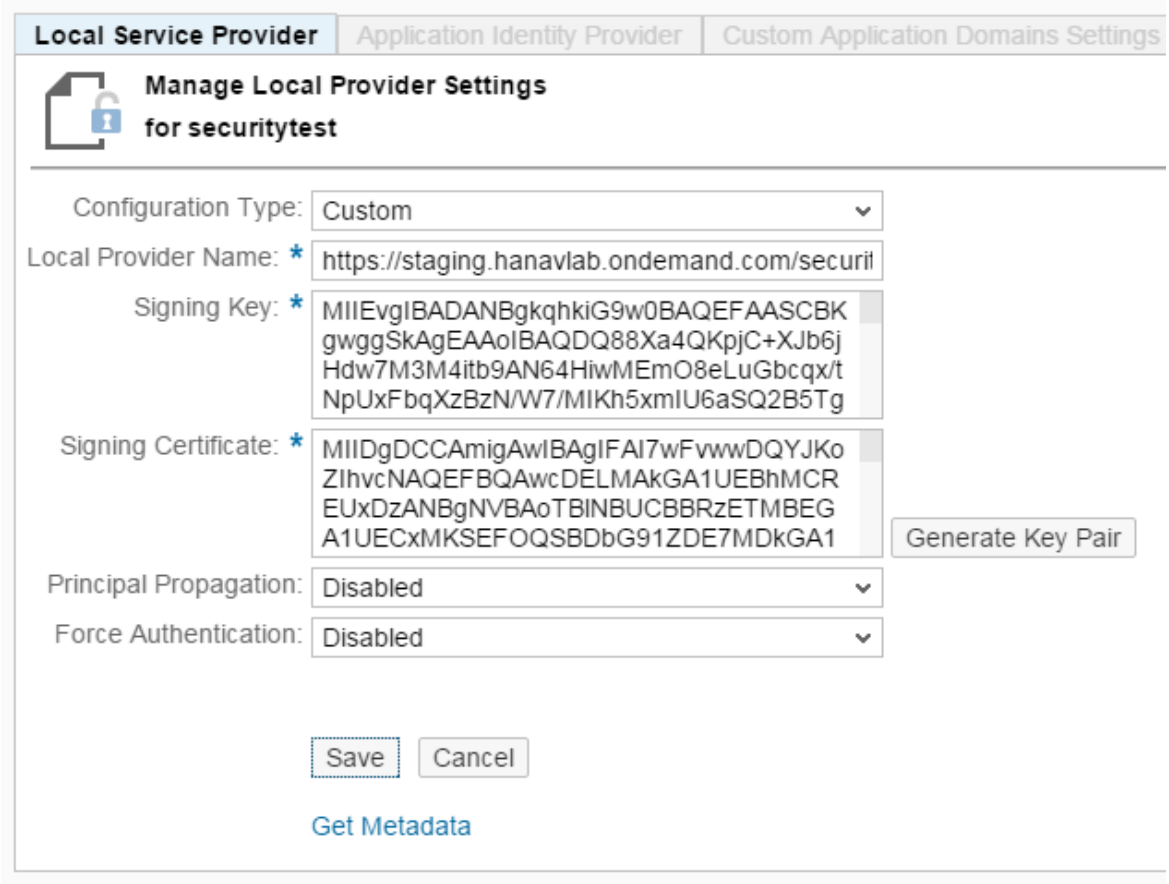
6. Double-click the server and open the *Users* tab page.
7. Define local test IdP users and their attributes. Exemplary data:



For more information about the Users editor, see Testing User Authentication on the Local Server [page 1678].

2. Configure the service provider of your account in SAP BTP

1. In a Web browser, open the cockpit and navigate to *Security* > *Trust* > *Local Service Provider*.
2. Choose *Edit*.
3. For *Configuration Type*, choose *Custom*.
4. Choose *Generate Key Pair* to generate a new signing key and self-signed certificate.
5. For the rest of the fields, leave the default values.
6. Choose *Save*.
7. Choose *Get Metadata* to download and save the SAML 2.0 metadata identifying your SAP BTP account as a service provider. You will have to import this metadata into the local test IdP to configure trust to SAP BTP in the procedure that follows.



3. (Optional) Configure the local IdP name

You need to configure your local IdP name if you want to use more than one local IdP. Default local IdP name: `localidp`.

1. In the Eclipse IDE, go to the already set up local server that will be used as local IdP.
2. In the `config_master/com.sap.core.jpaas.security.saml2.cfg/` folder, create a file named `local_idp.cfg`.
3. In the file, add a property:
   `localidp_name=<idpname you want to use>`
4. Restart the local server.

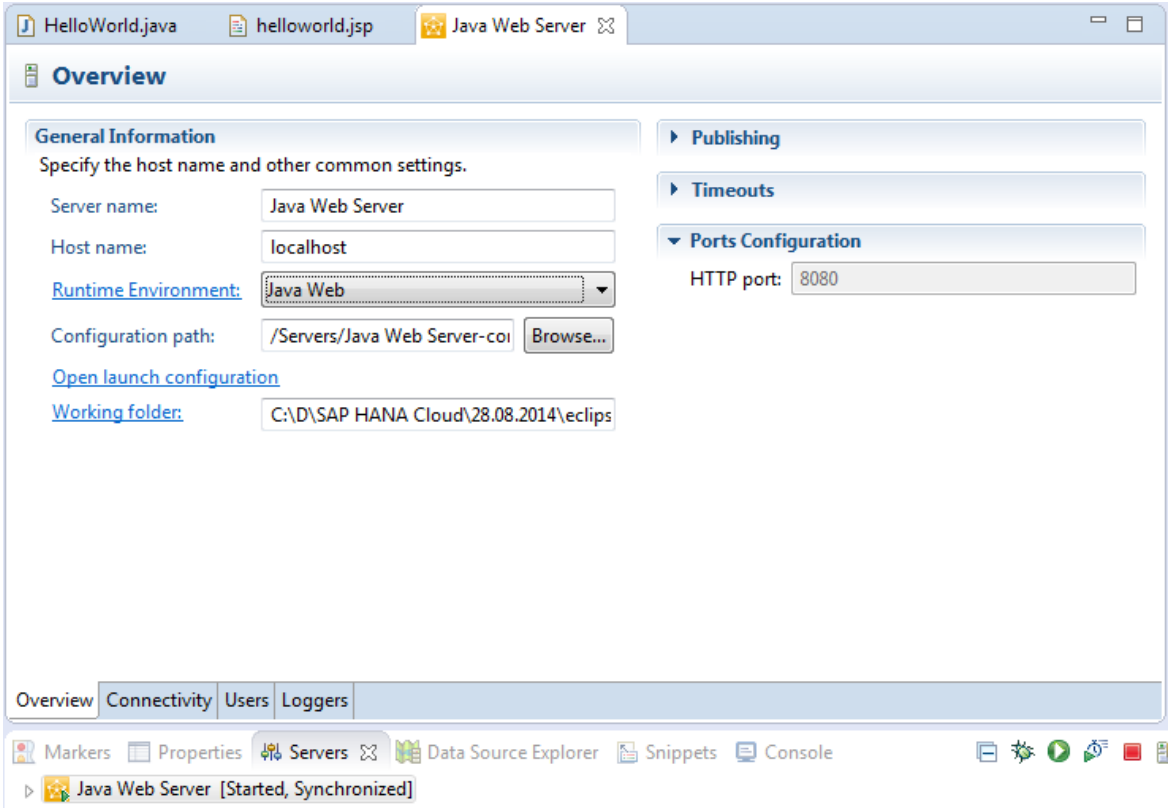
4. Configure trust on SAP BTP to the local test IdP

The trust settings on SAP BTP for the local test IdP are configured in the same way as with any other productive IdP.

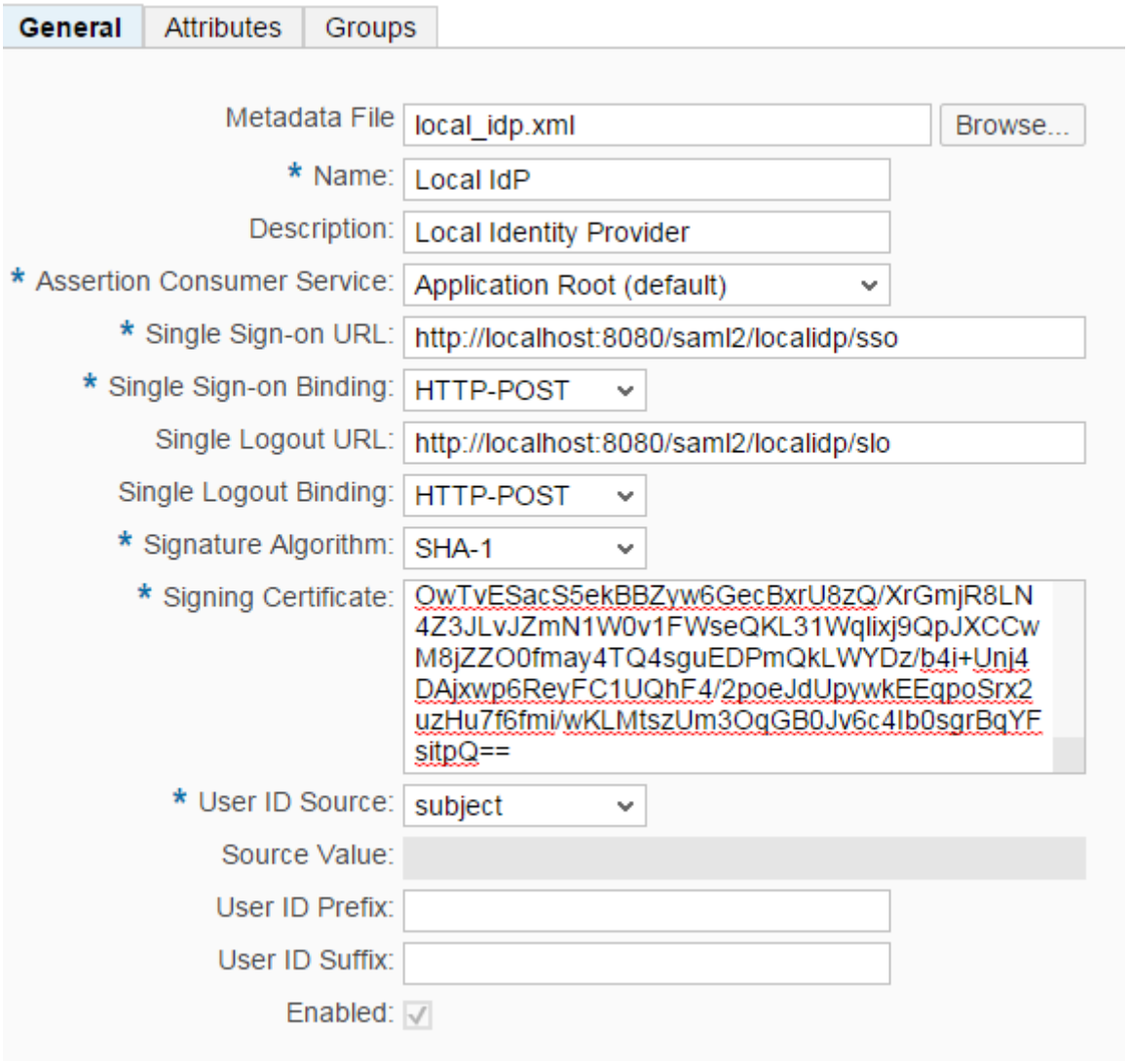
1. During the configuration, use the local test IdP metadata that can be requested under the following link: **`http://<idp_host>:<idp_port>/saml2/localidp/metadata`**,

where `<idp_host>` and `<idp_port>` are the local server host and port.

To find the `<idp_port>`, go to *Servers*, double click on the local server and choose *Overview* > *Ports Configuration*.



2. Configure trust on the cloud to the local test IdP.
    1. Open the cockpit in a Web browser, navigate to *Security* > *Trust* > *Application Identity Provider*, and then click *Add Trusted Identity Provider*.
    2. In *General* tab page, use the *Metadata File Browse* button to add the local test IdP metadata. All the needed values are filled in automatically.
    3. Choose *Save & Close*.



For more information, see Application Identity Provider [page 1720]

3. Configure the *User Attributes*.

Assertion-based attributes are used to define a mapping between attributes in the SAML assertion issued by the local test IdP and user attributes on the Cloud.

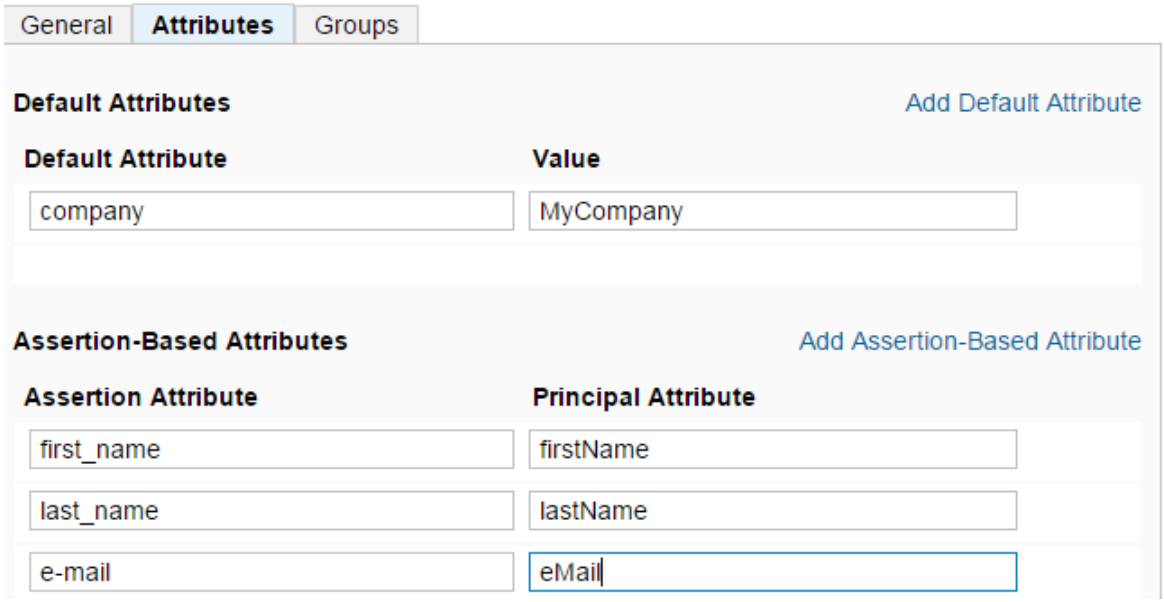
This allows you to essentially pass any attribute exposed by the local test IdP to an attribute used in your application in the cloud.

Define user attributes in the local test IdP by using the Eclipse IDE Users editor for SAP BTP as is described in Setting up the local test IdP.

To add an assertion-based attribute, proceed as follows:

1. Open the cockpit in a Web browser, navigate to *Security* > *Trust* > *Application Identity Provider*.
2. From the table, choose the entry *localidp*, open the *Attributes* tab page, and click on *Add Assertion-Based Attribute*.
3. In Assertion *Attribute*, enter the name of the attribute contained in the SAML 2.0 assertion issued by the local test IdP. These are the same user attributes you defined in the Eclipse IDE Users editor when setting the local test IdP.

4. In *Principal Attribute*, enter the name of the user attribute as referred in the tested application.



5. Choose *Save & Close*.

5. Generate self sign-key pair and certificate for the local test IdP (optional)

If an error occurs while requesting the IdP metadata and the metadata cannot be generated, you can do the following:

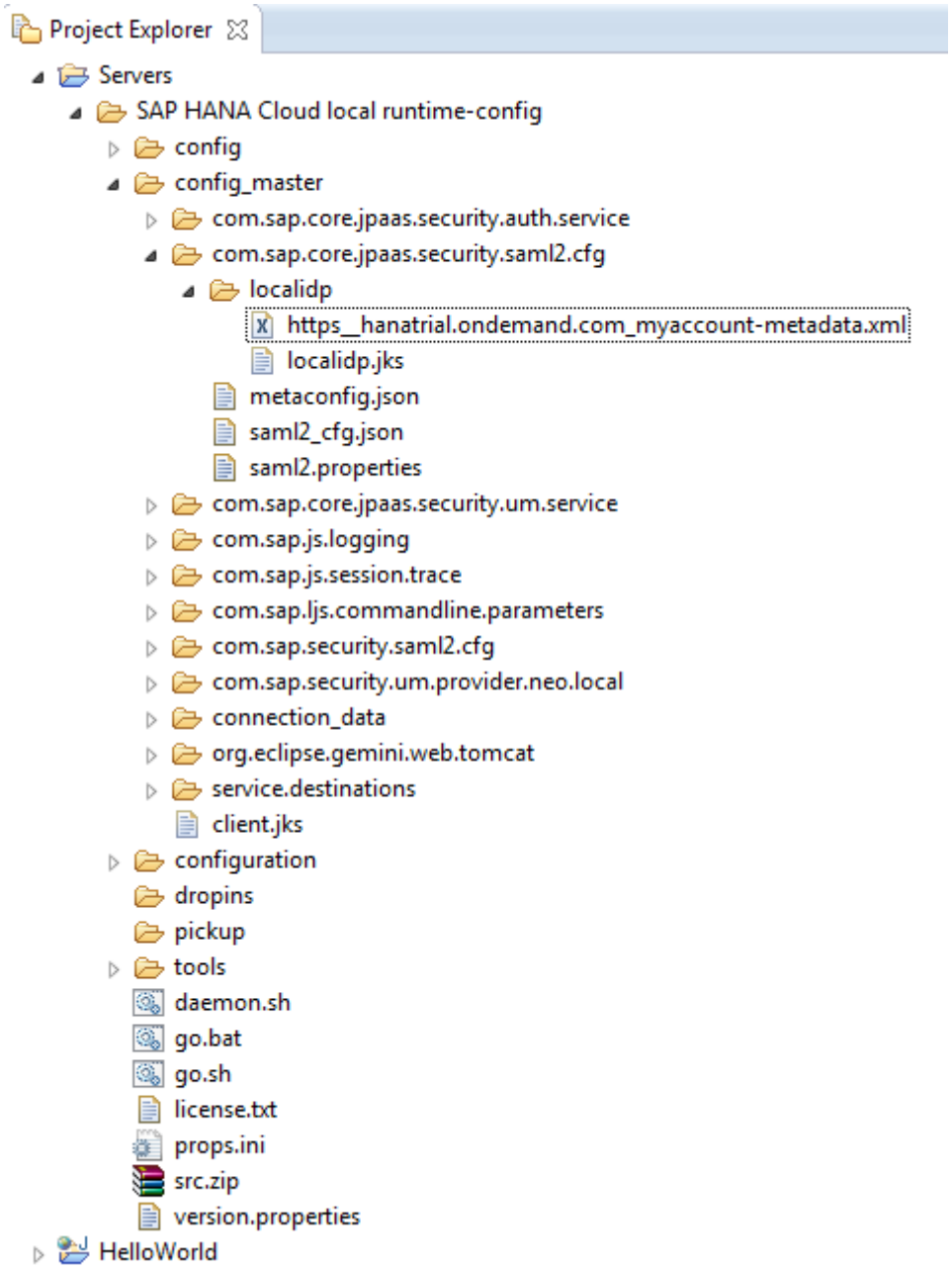
1. Generate a *localidp.jks* keyfile manually. The key and certificate are needed for signing the information that the local test IdP will exchange with SAP BTP.
2. Open the directory `<JAVA_HOME>/jre/bin/keytool`
3. Open a command line and execute the following command:

```
keytool -genkeypair -dname "CN=localidp" -keyalg "RSA" -validity 3650 -alias
localidp -storepass localidp -keypass localidp -keystore <fullpath_dir_name>
\localidp.jks
```

   where <fullpath_dir_name> is the directory path where the jks will be saved after the creation.
4. Under the *Server* directory, go to `config_master\com.sap.core.jpaas.security.saml2.cfg` and create a directory with name *localidp*.
5. Copy the *localidp.jks* file under *localidp* directory.

6. Configure trust on the local test IdP to SAP BTP

1. In the Eclipse IDE, go to the already set up local test IdP Server.
2. Copy the file with the metadata describing SAP BTP as a service provider under the local server directory `config_master/com.sap.core.jpaas.security.saml2.cfg/localidp`. To get this metadata, in the cockpit, choose *Security* > *Trust* > *Local Service Provider* > *Get Metadata*.



Result

You can now access your application, deployed on the cloud, and test it against the local test IdP and its defined users and attributes.

7.1.1.3 Security Configuration

When you have implemented security in your application, you need to perform a few configuration tasks using the Cockpit to enable the scenario to work successfully on SAP BTP.

Related Information

Managing Roles [page 1710]
Application Identity Provider [page 1720]

7.1.1.3.1 Managing Roles

In SAP BTP, you can use Java EE roles to define access to the application resources.

Context

Terms

Term	Description
Role	Roles allow you to diversify user access to application resources (role-based authorizations). **i Note** Role names are case sensitive.

Term	Description
Predefined roles	Predefined roles are ones defined in the *web.xml* of an application. See Authentication [page 1678]. After you deploy the application to SAP BTP, the role becomes visible in the Cockpit, and you can assign groups or individual users to that role. If you undeploy your application, these roles are removed. Predefined roles can be: • Shared - they are shared by default. A shared role is visible and accessible within all accounts subscribed to this application. • Restricted - an application administrator could restrict a shared role. A restricted role is visible and accessible only within the subaccount that deployed the application, and not to accounts subscribed to the application. **i Note** If you restrict a shared role, you hide it from visibility for new assignments from subscribed accounts but all existing assignments will continue to take effect.
Custom roles	Custom roles are ones defined using the Cockpit. Custom roles are interpreted in the same way as predefined roles at SAP BTP: they differ only in the way they are created, and in their scope. You can add custom roles to an application to configure additional access permissions to it without modifying the application's source code. Custom roles are visible and accessible only within the subaccount where they are created. That's why different accounts subscribed to the same application could have different custom roles.
User	Users are principals managed by identity providers (SAP ID service or others). **i Note** SAP BTP does not have a user database on its own. It cares to map the users authorized by identity providers to groups, and groups to roles. **i Note** When a user logs in, its roles are stored in the user's current browser session. They are **not** updated dynamically, and removed from there only if the session is terminated or invalidated. This means if you change the set of roles for a user currently logged, they will take effect only after logout or session invalidation.

Term	Description
Group	Groups are collections of roles that allow the definition of business-level functions within your sub-account. They are similar to the actual business roles existing in an organization, such as "manager", "employee", "external" and so on. They help you to get better alignment between technical Java EE roles and organizational roles. **i Note** Group names are case insensitive. For each identity provider (IdP) for your subaccount, you define a set of rules specifying the groups a user for this IdP belongs to. See Using a Custom Identity Provider [page 1720].

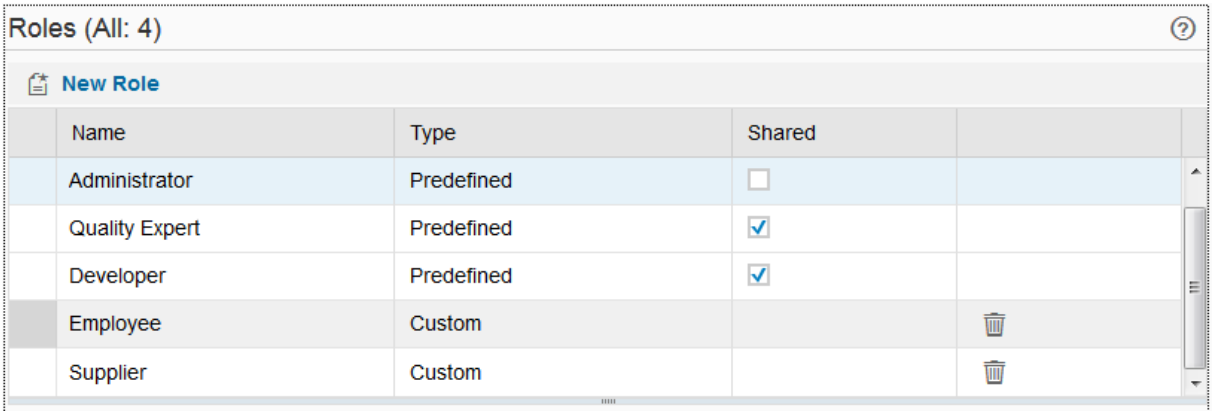
1. Define the Roles

Context

This can be done in two ways: using *predefined roles* in the web.xml at development time, or using *custom roles* in the UI.

→ Tip

If you need to do mass role or group assignment, to a very large number of users simultaneously, we recommend using the Authorization Management API instead of the cockpit UI. See Using Platform APIs [page 1149].



Procedure

- Predefined Roles

a. In the *web.xml* of the required application, define the roles authorized to access the application resources.

   See Authentication [page 1678].

b. Deploy the application to SAP BTP.

   See Deploying and Updating Applications [page 875].

c. Optionally, if you want to restrict the roles to the current application only, deselect the *Share* option for them in the Cockpit.

- Custom roles with applications from the same subaccount

  a. In the cockpit, go to the *Applications* > *Java Applications* section.
  b. Click the required application link.
  c. Enter the *Roles* section.
  d. Choose *New Role*.
  e. Type the role name and choose *OK*.

- Custom roles with applications subscribed from other accounts

  a. In the cockpit, go to the *Applications* > *Subscriptions* section.
  b. In the list of subscribed aplications, click the required application link.
  c. Enter the *Roles* section.
  d. Choose *New Role*.
  e. Type the role name and choose *OK*.

2. (Optional) Create Groups

Context

Groups allow you to easily manage the role assignments to collections of users instead of individual users.

Procedure

1. In the cockpit, go to the *Security* > *Authorizations* section for the subaccount.
2. Enter the *Groups* tab.
3. Choose *New Group*.
4. Enter the group name and choose *Save*.

3. Assign Users or Groups to the Roles

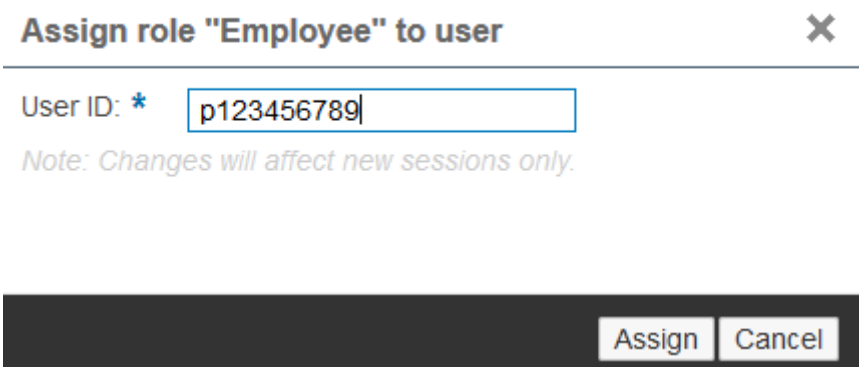
Context

You can assign individual users to the roles or, more conveniently, assign groups for collective role management.

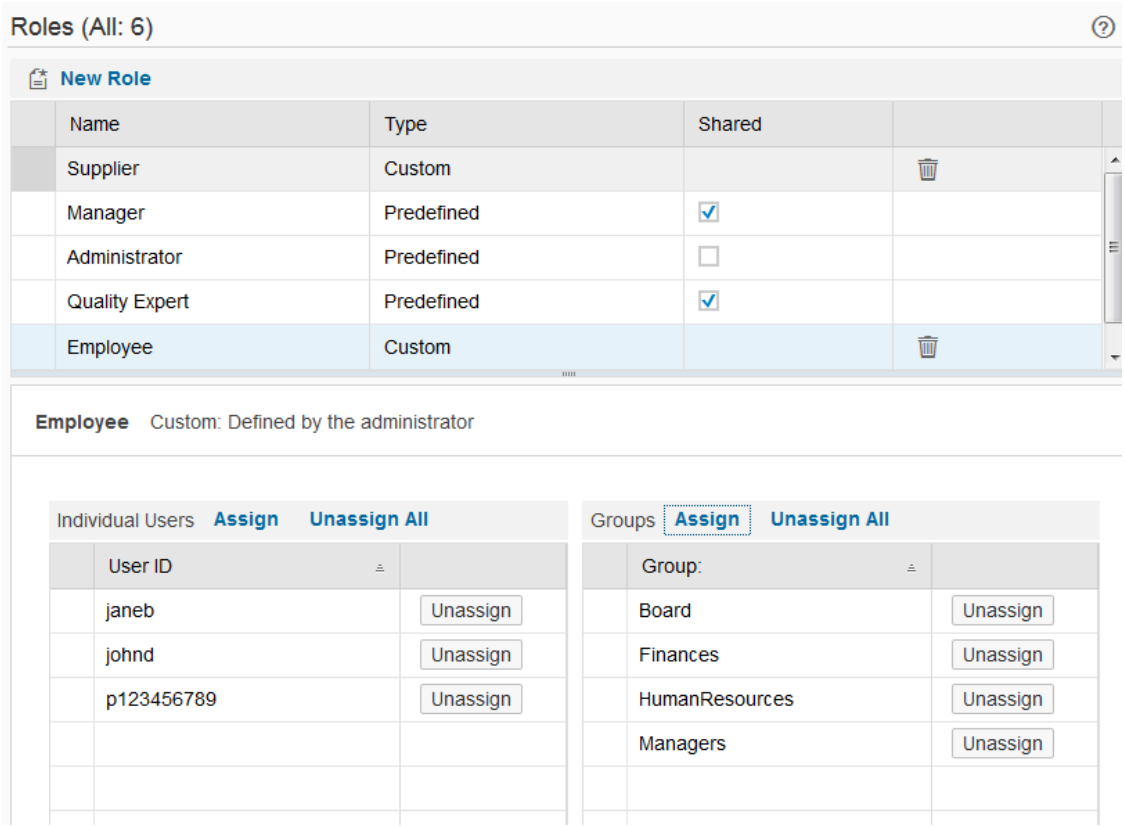
You can do it in either of the two ways: using the *Security* > *Roles* section for the application, or using the *Security* > *Authorizations* section for the subaccount.

Procedure

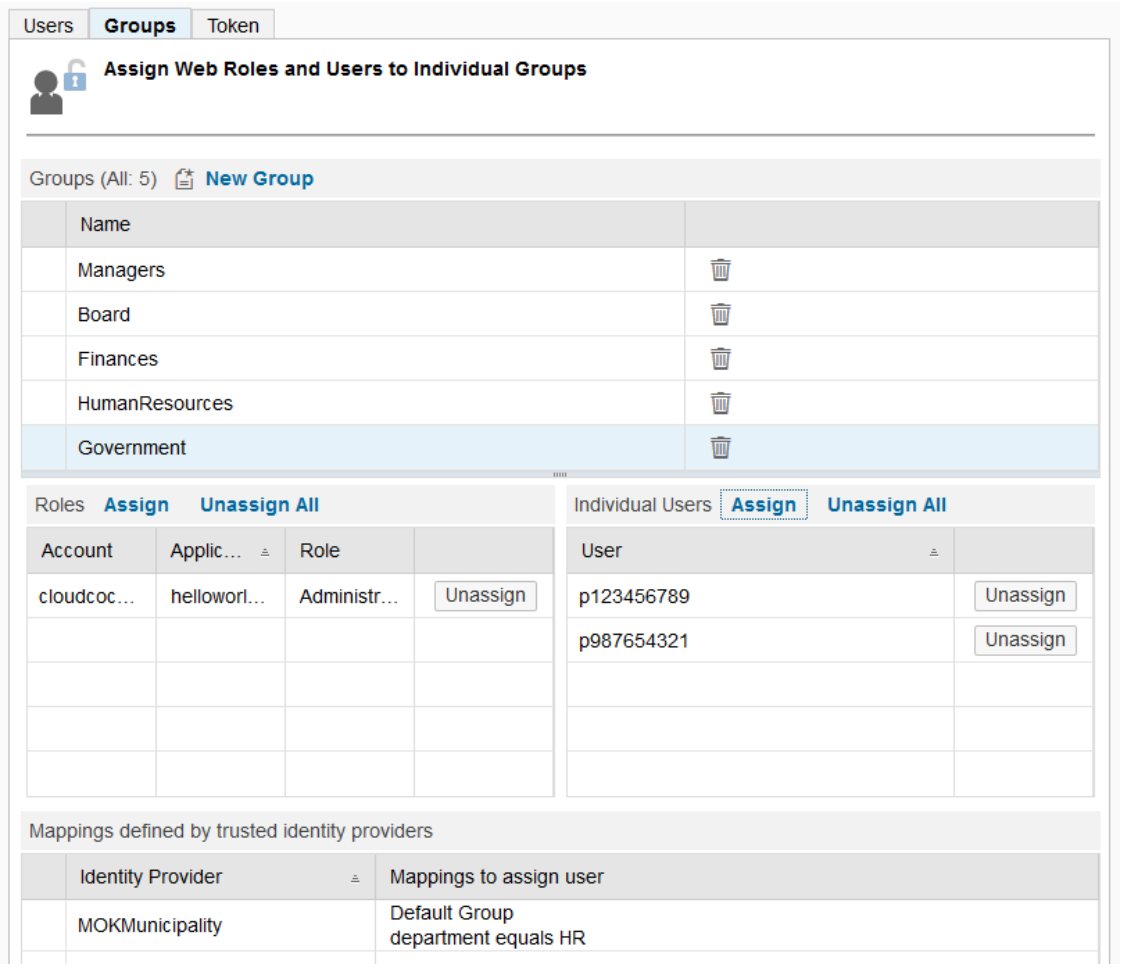
- Using the *Roles* section
  a. In the cockpit, go to the *Applications* > *Java Applications* section.
  b. Click the required application link.
  c. Enter the *Security* > *Roles* section.
  d. Select the role you want to manage assignments for.
  e. To assign a new user or group, choose *Assign* for the *Users* or *Groups* section respectively.
  f. Enter the user or group name.



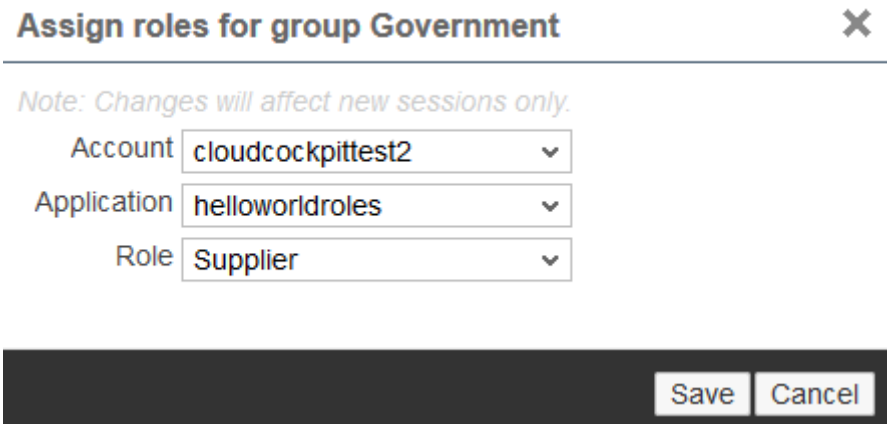
  g. Save the changes.



- Alternatively, you can do it using the *Authorizations* section for the subaccount.
  a. In the cockpit, go to the *Security* > *Authorizations* section for the subaccount.
  b. Enter the *Users* or *Groups* tab respectively.
  c. In the *User* or *Group* field respectively, enter the name of the required user or group. If you want to create a new user or group, simply assign the required roles from here.
  d. Choose *Show Assignments*. The table below shows all roles that are already assigned to this user or group.



e. If you are adding an individual user, choose the required application and the role the user will have. If you are adding a group, select the existing group from the list.



> → Tip
>
> You can use regular expressions to narrow the groups found.

f. Save the changes.

4. (If Using an Identity Provider) Define the Group-to-Role Mapping

Context

For each different IdP, you then define a set of rules specifying to which groups a user logged by this IdP belongs.

> **i Note**
>
> You must have defined groups in advance before you define default or assertion-based groups for this IdP.

Default groups are the groups all users logged by this IdP will have. For example, all users logged by the company IdP can belong to the group "*Internal*".

Assertion-based groups are groups determined by values of attributes in the SAML 2.0 assertion. For example, if the assertion contains the attribute "`contract=temporary`", you may want all such users to be added to the group "*TEMPORARY*".

Procedure

- Defining Default Groups
  a. In the cockpit, navigate to *Security* > *Authorizations* > *Groups*, and choose *Add Default Group*.
  b. From the dropdown list that appears, choose the required group.
- Defining Assertion-Based Groups
  a. In the cockpit, navigate to *Security* > *Authorizations* > *Groups*, and choose *Add Assertion-Based Group*. A new row appears and a new mapping rule is now being created.
  b. Enter the name of the group to which users will be mapped. Then define the rule for this mapping.
  c. In the first field of the *Mapping Rules* section, enter the SAML 2.0 assertion attribute name to be used as the mapping source. In other words, the value of this attribute will be compared with the value you specify (in the last field of *Mapping Rules*).
  d. Choose the comparison operator.

Equals	Choose *Equals* if you want the value of the SAML 2.0 assertion attribute to match exactly the string you specify. Note that if you want to use more sophisticated relations, such as "starts with" or "contains", you need to use the *Regular expression* option.

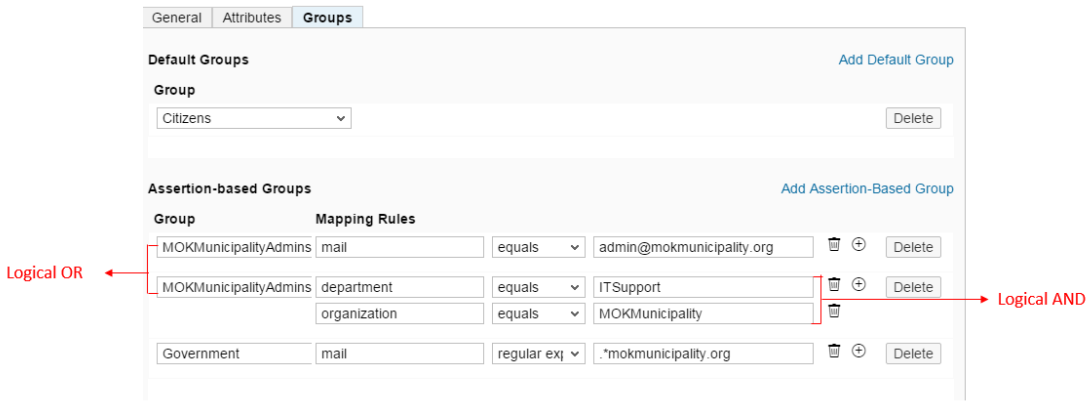
Regular expression	Choose *Regular expression* if you want to specify more sophisticated matching rules. You can use all regular expression rules described in the Java RegEx API . **Example 1**: You want to add authenticated SAP employees to group *Employees*. And SAP employees are users with e-mail address ending with *sap.com*. Hence, you choose the mapping rule to be *email*, matched using the following regular expression: *.*@sap.com$* **Example 2**: You want all users with name starting with admin to be added to group *Administrators*. Hence, you choose the mapping rule to be userid, matched using the following regular expression: *^(admin).**

e. In the last field of *Mapping Rules*, enter the value with which you compare the specified SAML 2.0 assertion attribute.
f. You can specify more than one mapping rule for a specific group. Use the plus button to add as many rules as required.

> **i Note**
>
> Adding a new rule binds it to the rest using a logical OR operator.

> **i Note**
>
> Adding a new subrule binds it to the rest of the subrules using a logical AND operator.

In the image below, all users logged by this IdP are added to the group *Government*. The users that have an arrtibute corresponding to their department name will also be assigned to the respective department groups.



When you open the *Groups* tab page of the *Authorizations* section, you can see the identity provider mappings for this group.

5. Test

Try to access the required application logging on with users with and without the required roles respectively.

(Optional) Configure Role Caching

Context

You may use the following steps to configure default role caching settings. This may be required if you have automated test procedures for role assignments in your applications. Tests may not work properly with the default subaccount settings.

The default role caching behavior is as follows:

If there are more than `com.sap.security.um.ratelimiter.cache.maximum_requests_per_user` (default is `20`) stateless requests within `com.sap.security.um.ratelimiter.cache.time_period` (default is `2 minutes`) with the same user to check the user permission, we cache hers roles for `com.sap.security.um.ratelimiter.cache.validity` (default is `5 minutes`). This means that new assignments/unassignments will not work for 5 minutes.

> → Tip
>
> You can take one of the following approaches:
>
> - Increase the time in which the requests are counted to more than the default 2 minutes.
> - Increase the number of requests – instead of the default 20, set 100 or 200, for example.

The table below shows the VM system properties available for configuring role caching:

VM Property	Description	Default Value
com.sap.security.um.ratelimiter.cache.maximum_user_entries	The maximum user entries stored in the role cache.	1000
com.sap.security.um.ratelimiter.cache.validity	The cache validity in time.	5 (in minutes)
com.sap.security.um.ratelimiter.cache.time_period	The time period for role caching.	2 (in minutes)
com.sap.security.um.ratelimiter.cache.maximum_requests_per_user	The maximum requests per user stored in the role cache.	20

Procedure

Set the required values to the required VM system properties as described in Configure VM Arguments [page 1598].

7.1.1.3.2 Application Identity Provider

The application identity provider supplies the user base for your applications. For example, you can use your corporate identity provider for your applications. This is called identity federation. SAP BTP supports Security Assertion Markup Language (SAML) 2.0 for identity federation.

Contents

- Prerequisites [page 1720]
- Configure the Local Service Provider [page 1721]
- (Optional) Using External Key and Certificate [page 1723]
- Configure Trust to the SAML Identity Provider [page 1724]
- Using Multiple Identity Providers [page 1732]

Prerequisites

- You have a key pair and certificate for signing the information you exchange with the IdP on behalf of SAP BTP. This ensures the privacy and integrity of the data exchanged. You can use your pre-generated ones or use the generation option in the cockpit.
- You have provided the IdP with the above certificate. This allows the IdP administrator to configure its trust settings.
- You have the IdP signing certificate to enable you to configure the cloud trust settings.
- You have negotiated with the IdP administrator which information the SAML 2.0 assertion will contain for each user. For example, this could be a first name, last name, company, position, or an e-mail.
- You know the authorizations and attributes the users logged by this IdP need to have on SAP BTP.

> → Tip
>
> You can configure your SAP BTP account for identity federation with more than one identity provider. In such case, make sure all user identities are unique across all identity providers, and no user is available in more than one identity provider. Otherwise, this could lead to wrong assignment of security roles at SAP BTP.

7.1.1.3.2.1 Configure the Local Service Provider

Your SAP BTP subaccount is the local service provider in the SAML communication. Configure signing keys, certificates, and other trust settings.

Context

For more information, see Security Assertion Markup Language (SAML) 2.0 protocol specification.

> → Tip
>
> Each SAP BTP subaccount is a separate service provider. If you need each of your applications to be represented by its own service provider, you must create and use a separate subaccount for each application. See Create a Subaccount.

> i Note
>
> In this documentation and SAP BTP user interface, we use the term **local service provider** to describe the SAP BTP subaccount as a service provider in the SAML 2.0 communication.

You need to configure how the local service provider communicates with the identity provider. This includes, for example, setting a signing key and certificate to verify the service provider's identity and encrypt data. You can use the configuration settings described in the table that follows.

Local Service Provider Configuration	Description	When to Use
Default	The local provider's own trust settings will inherit the SAP BTP default configuration (which is trust to **SAP ID service**).	For testing and exploring the scenario
None	The local provider will have no trust settings, and it will not participate in any identity federation scenario.	For disabling identity federation for your account
Custom	The local provider settings will have a specific configuration, different from the default configuration for SAP BTP.	For identity federation with a corporate identity provider or Identity Authentication tenant

In addition, you can configure the following local service provider settings:

Local Service Provider Configuration (Additional)	Description
Principal Propagation	If you set it to *Enabled*, you enable applications to propagate principal information to each other. Choose this value if you want to enable application-to-application single sign-on. Otherwise, set this option to *Disabled*.
Force authentication	If you set it to *Enabled*, you enable force authentication for your application (despite SSO, users will have to re-authenticate each time they access it). Otherwise, set this option to *Disabled*.

Procedure

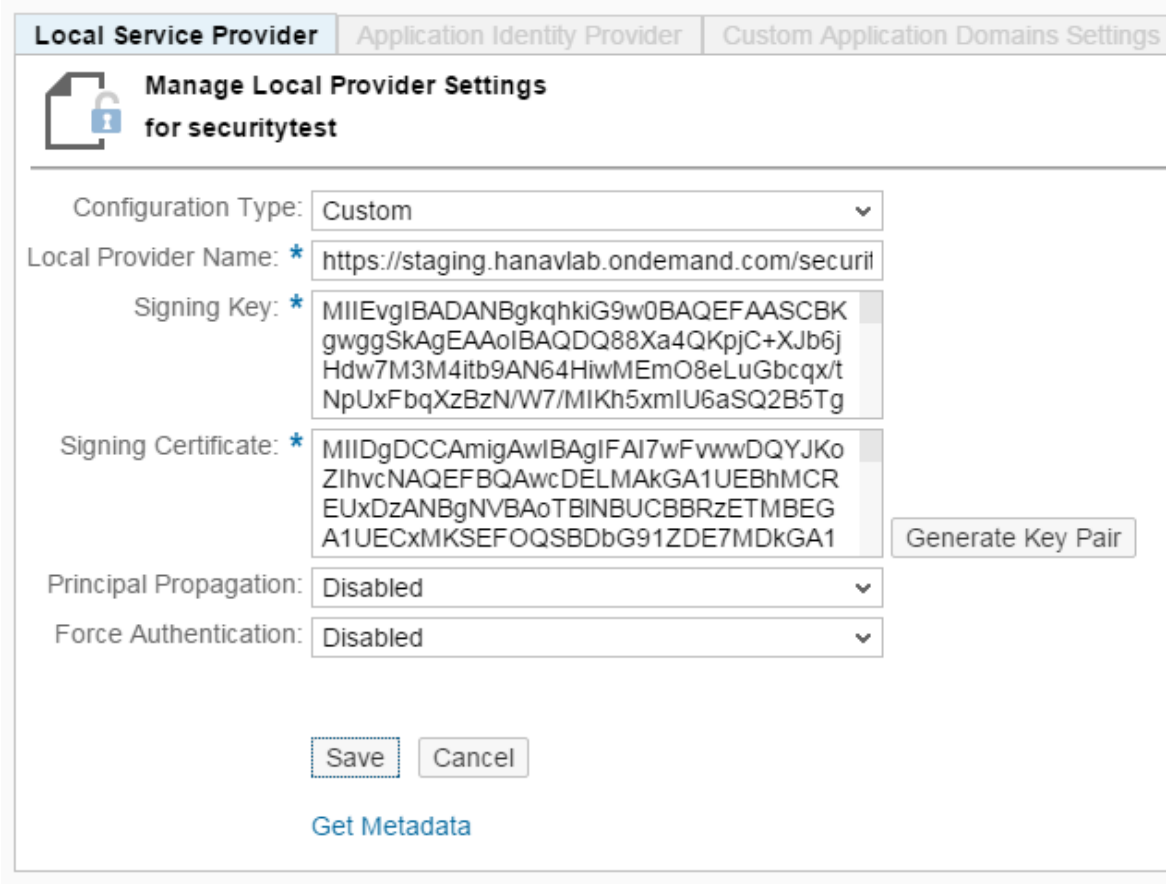
1. In your Web browser, log on to the SAP BTP cockpit, and select an account.

   Make sure that you have selected the relevant global account to be able to select the right account.
2. Choose the *Security* > *Trust* section.
3. Choose the *Local Service Provider* tab.
4. Choose *Edit*.
5. Choose *Custom* as *Configuration Type*.
6. Enter the local provider name.

   > **i Note**
   >
   > It is recommended to use a URI as the local provider name.
7. In *Signing Key* and *Signing Certificate*, place the Base64-encoded signing key and certificate. You can use one generated with the SAP BTP cockpit (using the *Generate Key Pair* button) or externally generated one.

   > **i Note**
   >
   > Certificates generated using the SAP BTP cockpit have validity of 10 years. If you want your identifying certificate to have different validity, generate the key and certificate pair using an external tool, and copy the contents in the *Signing Key* and *Signing Certificate* fields respectively in the SAP BTP cockpit.

   > **i Note**
   >
   > For more information how to use an externally generated key and certificate pair, see (Optional) Using External Key and Certificate [page 1723].
8. Choose the required value of the *Principal Propagation* and *Force authentication* option.
9. Save the changes.
10. Choose *Get Metadata* to download the SAML 2.0 metadata describing SAP BTP as a service provider. You will have to import this metadata into the IdP to configure trust to SAP BTP.



7.1.1.3.2.2 (Optional) Using External Key and Certificate

If you want to use for the local service provider a signing key and certificate generated using an **external tool** (such as OpenSSL), use the following guidelines:

- Use RSA as a signing algorithm for the private key
- Convert the private key file into the unencrypted PKCS#8 format
- Strip off the tags `--BEGIN PRIVATE KEY--` and `--END PRIVATE KEY--` from the private key
- Strip off the tags `--BEGIN CERTIFICATE--` and `--END CERTIFICATE--` from the certificate.

Example

You want to use OpenSSL as a tool for key pair generation.

First, create the key pair using the following command:

```
openssl req -x509 -nodes -days 365 -sha256 -subj
"/CN=https:\/\/SAP BTP host>\/<your account name>" -newkey rsa:2048
-keyout spkey.pem -out spcert.pem
```

In the command above, replace `<SAP BTP host>` and `<your account name>` accordingly. For more information about the SAP BTP hosts, see Regions and Hosts Available for the Neo Environment [page 17].

As a result, OpenSSL generates two files in your current folder: `spkey.pem` (your private key) and `spcert.pem` (a self-signed signing certificate).

> **i Note**
>
> If you need the certificate to be signed by a certificate authority (CA), you need to proceed with a few more steps:
>
> 1. Generate a certificate signing request (CSR) by executing the following command in the folder of your `spkey.pem`:
>
>    ```
>    openssl req -new -sha256 -key spkey.pem -out spkey.csr
>    ```
>
>   OpenSSL will ask you to enter the fields of the CSR. For the `Common Name` field, we recommend that you use the following format:
>   `https:\/\/<SAP BTP host>\/<your account name>`.
>   As a result, OpenSSL generates one more file in your current folder: `spkey.csr` (the CSR for your key/certificate pair).
> 2. Send the `spkey.csr` to your CA to get it signed.
>   The CA returns the signed certificate. You can use that certificate in the steps below.

Convert the private key file `spkey.pem` into the unencrypted PKCS#8 format using the following command:

```
openssl pkcs8 -nocrypt -topk8 -inform PEM -outform PEM -in spkey.pem -out
spkey.pk8
```

Now open the file `spkey.pk8` in a text editor and copy all contents except for the tags `--BEGIN PRIVATE KEY--`, `--END PRIVATE KEY--` into the *Signing Key* text field in the cockpit. Then open the file `spcert.pem` in a text editor and copy all contents except for the tags `--BEGIN CERTIFICATE--` and `--END CERTIFICATE --` into the *Signing Certificate* text field in the cockpit.

After clicking *Save* you should get a message that you can proceed with the configuring of your trusted identity provider settings.

7.1.1.3.2.3 Configure Trust to the SAML Identity Provider

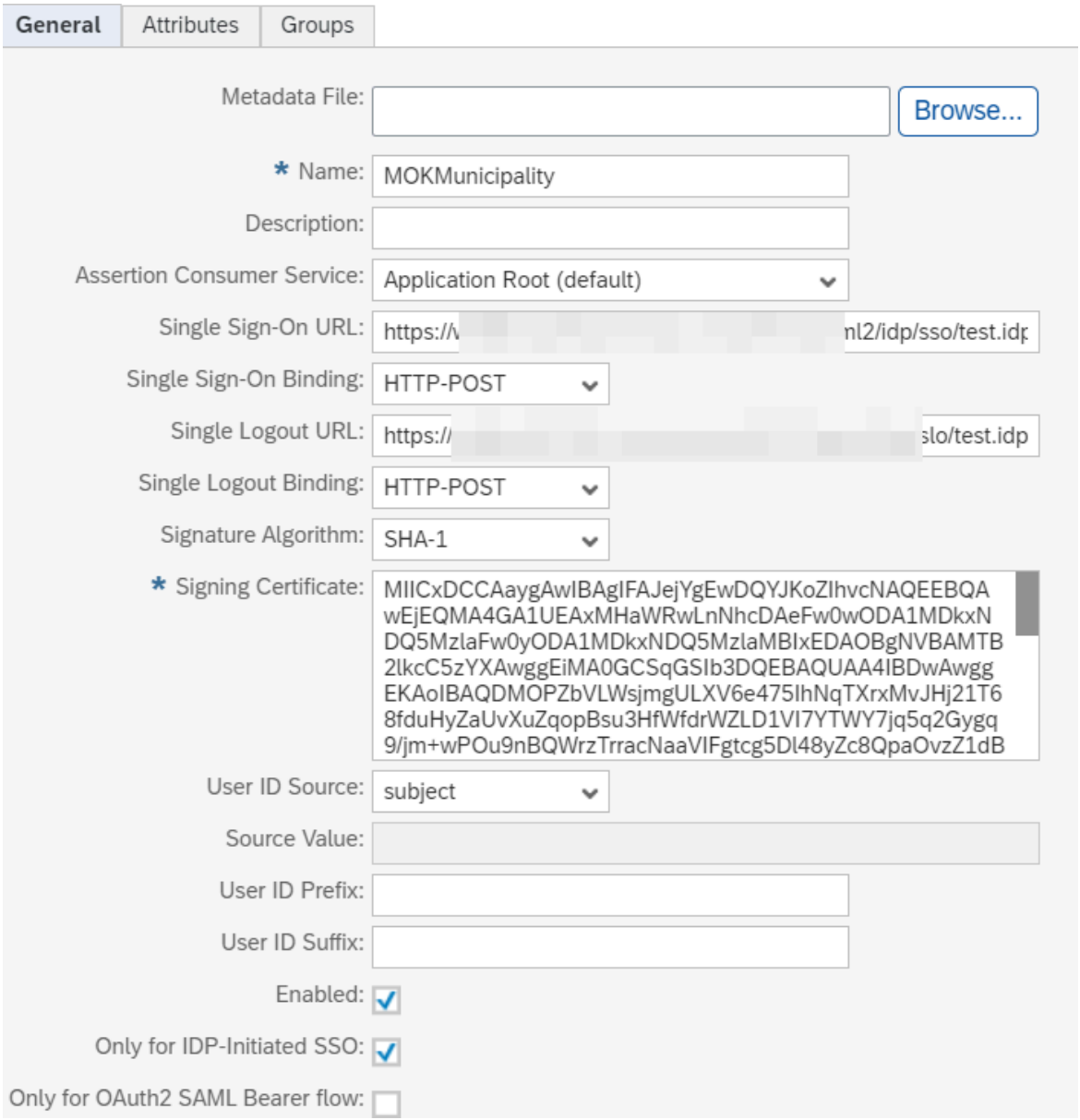
Context

> **i Note**
>
> To benefit from fully-featured identity federation with SAML identity providers, you need to have chosen the *Custom* configuration type in the *Local Service Provider* section.
>
> For *Default* configuration type, you have non-editable trust to **SAP ID Service as default identity provider**. You can add other identity providers but they can be used for **IdP-initiated single sign-on (SSO)** only.

For *None*, you don't have any trust settings.

Procedure

1. In the SAP BTP cockpit, navigate to the required SAP BTP subaccount. See Navigate in the Cockpit.
2. Enter the *Security* > *Trust* > *Application Identity Provider* section.
3. Click *Add Trusted Identity Provider*.
4. In the *General* tab, upload the IdP metadata xml file or manually enter he communication settings negotiated between SAP BTP and the IdP. On IdP metadata upload, the fields are populated with the parsed data from the XML file. The minimum configuration is to complete all required fields.



Field	Description
Metadata File	The metadata XML file of the identity provider.
Name	The entity ID of the IdP, also known as the issuer.
Description	A short description of the IdP.
Assertion Consumer Service	The SAP BTP endpoint type (application root or assertion consumer service). The IdP will send the SAML assertion to that endpoint. In the common case, select *Application Root* as value. If you have an identity provider that would not send the SAML assertion to unknown URLs to them, select the *Assertion Consumer Service* option. This is the case with Microsoft ADFS, for example.
Single Sign-on URL	The IdP's endpoint (URL) to which the SP's authentication request will be sent.
Single Sign-on Binding	The SAML-specified HTTP binding used by the SP to send the authentication request.
Single Logout URL	The IdP's endpoint (URL) to which the SP's logout request will be sent. **i Note** If there is no single logout (SLO) end point specified, no request to the IdP SLO point will be sent, and only the local session will be invalidated.
Signature Algorithm	The cryptographic algorithm used to compute the digest of the digital signatures in the SAML protocol messages.
Signing Certificate	The X.509 certificate used by the IdP to digitally sign the SAML protocol messages.
User ID Source	Location in the SAML assertion from where the user's unique name (ID) is taken when logging into the Cloud. If you choose subject, this is taken from the name identifier in the assertions's subject (<saml:Subject>) element. If you choose attribute, the user's name is taken from an SAML attribute in the assertion.
Source Value	Name of the SAML attribute that defines the user ID on the cloud.
User ID Prefix	An optional prefix added to the user ID on the cloud.
User ID Suffix	An optional suffix appended to the user ID on the cloud.
Enabled	If an IdP is enabled, it can be used for authentication by the applications in this subaccount. Otherwise, it cannot be used. Only the SAML assertions coming from it will be validated by SAP BTP.

Field	Description
	Note If nothing else is specified, the default IdP is used for authentication. Alternatively, you can use a different IdP using a URL parameter. See Using Multiple Identity Providers [page 1732].
Only for IDP-initiated SSO	If this checkbox is marked, this identity provider can be used only for IdP-initiated single sign-on scenarios. The applications deployed at SAP BTP cannot use it for user authentication from their login pages, for example. Only users coming from links to the application at the IdP side will be able to authenticate. **Note** When you add a new application identity provider and you have selected *Default* configuration type in the *Local Service Provider* section, this checkbox is always marked. This means that SAP ID Service (accounts.sap.com) will be used for authentication when accessing applications/services on SAP BTP, and the additional application identity provider can be used only for IDP-initiated SSO.
Only for OAuth2 SAML Bearer flow	The IdP will only be used to validate SAML Assertions received via the OAuth SAML Bearer Flow. This allows a more fine-granular and secure control of which IdPs are allowed during login.

5. In the *Attributes* tab, configure the user attribute mappings for this identity provider.

   User attributes can contain any other information in addition to the user ID.

   Default attributes are user attributes that all users logged by this IdP will have. For example, if we know that "*My IdP*" is used to authenticate users from *MyCompany*, we can set a default user attribute for that IdP "**`company=MyCompany`**".

   To add a default attribute, proceed as follows:

   1. On the *Attributes* tab page, choose *Add Default Attribute*.
   2. Enter the attribute name and attribute value in the respective fields.

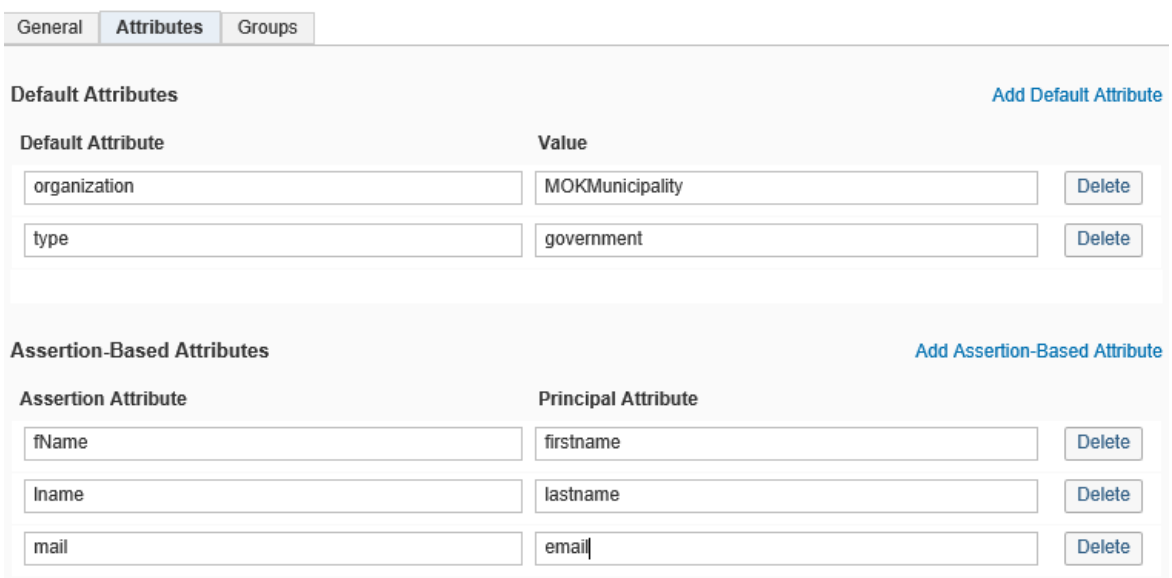
   Assertion-based attributes define a mapping between user attributes sent by the identity provider (in the SAML assertion) and user attributes consumed by applications on SAP BTP (**principal attributes**). This allows you to easily map the user information sent by the IdP to the format required by your application without having to change your application code. For example, the IdP sends the first name and last name user information in attributes named `first_name` and `last_name`. You, on the other hand, have a cloud application that retrieves user attributes named `firstName` and `lastName`. You need to define the

relevant mapping in the *Assertion-Based Attributes* section so the application uses the information from that identity provider properly.

> **i Note**
>
> - There are no default mappings of assertion attributes to user attributes. You need to define those if you need them.
> - The attributes are case sensitive.
> - You can specify that all assertion attributes will be mapped to the corresponding principal attributes without a change, by specifying mapping `*` to `*`.
> - SAML assertions larger than 25K are **not** supported.
> - We recommend that you avoid sending from the IdP side unnecessary user attributes (the same applies also for unnecessary groups mapping) as assertion attributes. Too many assertion attributes will result in a very long SAML assertion, which may put unnecessary load on communication (and potentially result in errors). Send only the user attributes that your cloud applications will really need.

To add an assertion-based attribute, proceed as follows:

1. On the *Attributes* tab page, choose *Add Assertion-Based Attribute*.
2. In *Assertion Attribute*, enter the name of the attribute contained in the SAML 2.0 assertion issued by the IdP. When this IdP logs a user on SAP BTP, the value of this attribute is mapped as the value for the specified user attribute (*Principal Attribute*).
3. In *Principal Attribute*, enter the name of the user attribute on SAP BTP.



In the screenshot above, all users authenticated by this IdP will have an attribute *organization="MOKMunicipality"* and *type="Government"*. In addition, several attributes (corresponding to first name, last name and e-mail) from the SAML assertion will also be added to authenticated users. Note that those attribute names provided in the assertion by the IdP are different from the principal attributes, which are the attributes used by the cloud applications.

For more information about using user attributes in your application, see Authentication [page 1678].

6. In the *Groups* tab, configure the groups associated with this IdP's users.

Groups that you define on the cloud are later mapped to Java EE application *roles*. As specified in Java EE, in the *web.xml*, you define the roles authorized to access a protected resource in your application. You therefore define the groups that exist there and the roles to which each group is mapped via the *Groups* tab in the SAP BTP cockpit. For each different IdP, you then define a set of rules specifying to which groups a user logged by this IdP belongs.

For more information about configuring groups, see Managing Groups and Roles [page 1710].

> **i Note**
>
> You must have defined groups in advance before you define default or assertion-based groups for this IdP.

Default groups are the groups all users logged by this IdP will have. For example, all users logged by the company IdP can belong to the group "*Internal*".

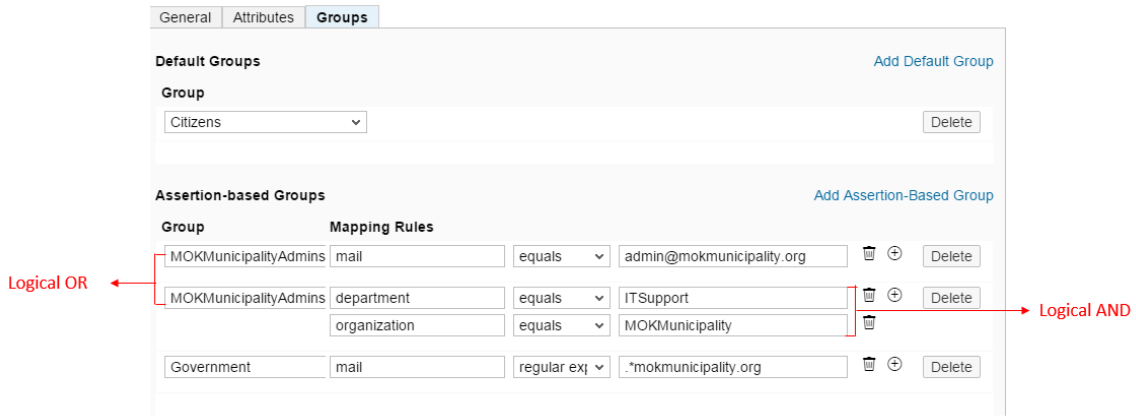
To add a default group, proceed as follows:

1. On the *Groups* tab page, choose *Add Default Group*.
2. From the dropdown list that appears, choose the required group.

Assertion-based groups are groups determined by values of attributes in the SAML 2.0 assertion. For example, if the assertion contains the attribute "`contract=temporary`", you may want all such users to be added to the group "*TEMPORARY*".

To add an assertion-based group, proceed as follows:

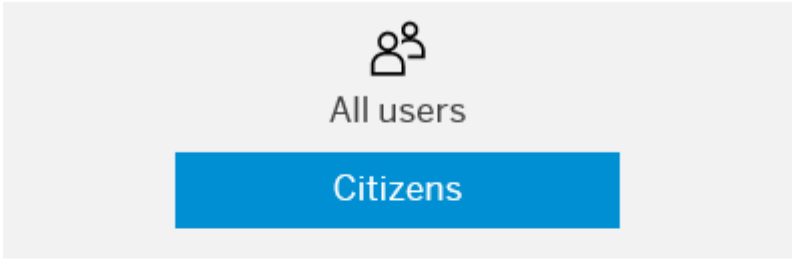
1. On the *GROUPS* tab page, choose *Add Assertion-Based Group*. A new row appears and a new mapping rule is now being created.
2. Enter the name of the group to which users will be mapped. Then define the rule for this mapping.
3. In the first field of the *Mapping Rules* section, enter the SAML 2.0 assertion attribute name to be used as the mapping source. In other words, the value of this attribute will be compared with the value you specify (in the last field of *Mapping Rules*).
4. Choose the comparison operator.
   - Choose *Equals* if you want the value of the SAML 2.0 assertion attribute to match exactly the string you specify. Note that if you want to use more sophisticated relations, such as "starts with" or "contains", you need to use the *Regular expression* option.
   - Choose *Regular expression* if you want to specify more sophisticated matching rules. You can use all regular expression rules described in the Java RegEx API .
     Example 1: You want to add authenticated SAP employees to group *Employees*. And SAP employees are users with e-mail address ending with *sap.com*. Hence, you choose the mapping rule to be *email*, matched using the following regular expression:
     .@sap.com$*
     Example 2: You want all users with name starting with admin to be added to group *Administrators*. Hence, you choose the mapping rule to be userid, matched using the following regular expression:
     *^(admin).**
5. In the last field of *Mapping Rules*, enter the value with which you compare the specified SAML 2.0 assertion attribute.
6. You can specify more than one mapping rule for a specific group. Use the plus button to add as many rules as required. In this case, mapping is based on a logical AND operation for all rules, that is, if one of your rules applies, the user is added to the group.

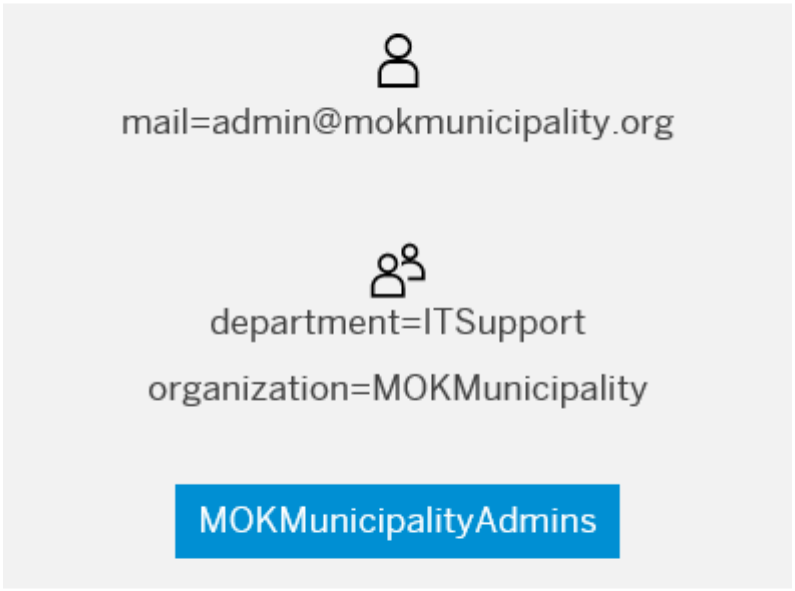


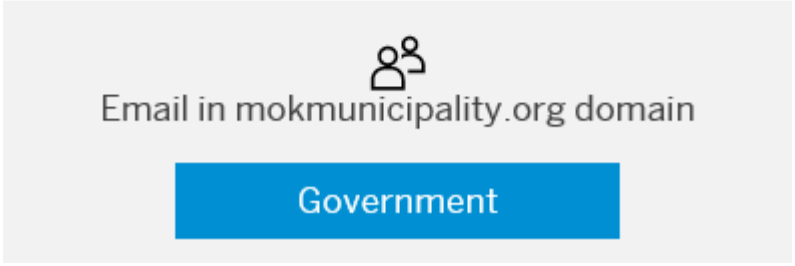
In the image above, all users logged by this IdP are added to the group *Citizens*.

All users from the *ITSupport* department (of organization MOKMunicipality) and the user with e-mail *admin@mokmunicipality.org* are added to group *MOKMunicipalityAdmins* for this subaccount. The rest of the employees at MOKMunicipality (having an e-mail address in the mokmunicipality.org domain) are assigned to group *Government*.

You can see the group assignments visualized in the graphic below.


All users
Citizens


mail=admin@mokmunicipality.org
department=ITSupport
organization=MOKMunicipality
MOKMunicipalityAdmins


Email in mokmunicipality.org domain
Government

7.1.1.3.2.4 Using Multiple Identity Providers

You may need to use a different identity provider (IdP) for each security scenario. For example, one IdP for user authentication, another one for IdP-initiated single sign-on (SSO), and a third one for OAuth 2.0 SAML Bearer flow.

Procedure

1. In the SAP BTP cockpit, configure trust with all required identity providers for your scenarios. See Configure Trust to the SAML Identity Provider [page 1724].

   One of the identity providers configured for the subaccount is *the default* one. This is the identity provider that will be used for user authentication. All the rest can be used either for IdP-initiated SSO, or for OAuth 2.0 SAML Bearer flow. Mark the respective option when registering the identity provider:

Field	Description
Only for IDP-initiated SSO	If this checkbox is marked, this identity provider can be used only for IdP-initiated single sign-on scenarios. The applications deployed at SAP BTP cannot use it for user authentication from their login pages, for example. Only users coming from links to the application at the IdP side will be able to authenticate. **i Note** When you add a new application identity provider and you have selected *Default* configuration type in the *Local Service Provider* section, this checkbox is always marked. This means that SAP ID Service (accounts.sap.com) will be used for authentication when accessing applications/services on SAP BTP, and the additional application identity provider can be used only for IDP-initiated SSO.
Only for OAuth2 SAML Bearer flow	The IdP will only be used to validate SAML Assertions received via the OAuth SAML Bearer Flow. This allows a more fine-granular and secure control of which IdPs are allowed during login.

2. In your application, request the identity provider you need (for IdP-initiated SSO or OAuth 2.0 SAML Bearer flow) using a special request parameter `saml2idp` with value the desired IdP name. For example:.

   ```
   https://<app name>.hana.ondemand.com/index.jsp?saml2idp=<idp name>
   ```

3. (For IdP-initiated SSO only) On the identity provider side, use the following service provider names to configure trust with your SAP BTP subaccount:

Technical Key of Region	Service Provider Name
neo-eu1	https://netweaver.ondemand.com
neo-eu2	https://eu2.hana.ondemand.com/
neo-eu3	https://eu3.hana.ondemand.com
neo-us1	https://us1.hana.ondemand.com/
neo-us2	https://us2.hana.ondemand.com
neo-us3	https://us3.hana.ondemand.com
neo-us4	https://us4.hana.ondemand.com
neo-ap1	ap1.hana.ondemand.com
neo-ap2	https://ap2.hana.ondemand.com
neo-jp1	https://jp1.hana.ondemand.com
neo-jp2	https://jp2.hana.ondemand.com
neo-cn1	https://cn1.hana.ondemand.com
neo-cn2	https://cn2.hana.ondemand.com
neo-ru1	https://ru1.hana.ondemand.com
neo-br1	https://br1.hana.ondemand.com
neo-br2	https://br2.hana.ondemand.com
neo-ae1	https://ae1.hana.ondemand.com
CA1	https://ca1.hana.ondemand.com

See Regions in the Neo Environment [page 16].

7.1.1.3.3 Identity Authentication Tenant as an Application Identity Provider

You can register a tenant for Identity Authentication service as an identity provider for your subaccount.

Prerequisites

- You have defined service provider settings for the SAP BTP subaccount. See Configure the Local Service Provider [page 1721].
- You have chosen a custom local provider configuration type for this subaccount (using *Cockpit* > *Trust* > *Local Service Provider* > *Configuration Type* > *Custom*)

Context

Identity Authentication service provides identity management for SAP BTP applications. You can register a tenant for Identity Authentication service as an identity provider for the applications in your SAP BTP subaccount.

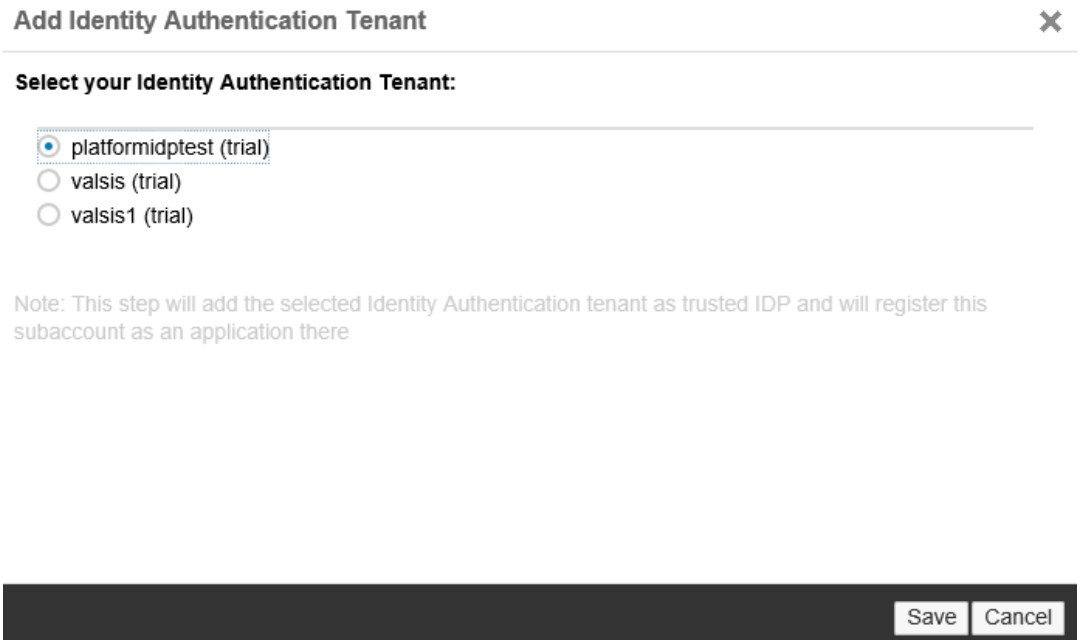
> **i Note**
>
> If you add a tenant for Identity Authentication service already configured for trust with the same service provider name, the existing trust configuration on the tenant for Identity Authentication service side will be updated. If you add a tenant for Identity Authentication configured for trust with SAP BTP with a different service provider name, a new trust configuration will be created on the tenant for Identity Authentication service side.

> **i Note**
>
> When you remove a tenant for Identity Authentication service as trusted identity provider, the relevant service provider configuration in the Identity Authentication tenant is preserved.

Procedure

1. In the SAP BTP cockpit, navigate to the required SAP BTP subaccount. See Navigate in the Cockpit.
2. Choose the *Security* > *Trust* section.
3. Choose the *Application Identity Provider* tab page. Proceed here depending on one of the following cases:
   - You have a tenant for Identity Authentication service registered for your current SAP customer user (s-user). You want to add the tenant as an identity provider.
     1. Click *Add Identity Authentication Tenant*.

2. Choose the required Identity Authentication tenant and save the changes.



In this case, the trust will be established automatically upon registration on both the SAP BTP and the tenant for Identity Authentication service side. See Initial Setup (Identity Authentication)

- You want to add a tenant for Identity Authentication service not related to your SAP user.
  In this case, you need to register the tenant for Identity Authentication service as described in Application Identity Provider [page 1720]. In addition, configure trust on the tenant as described in Configure Trust (Identity Authentication).

Results

The tenant for Identity Authentication appears in the list of SAML identity providers. You can now administrate further the Identity Authentication tenant by opening Identity Authentication Admin Console (hover over the registered tenant for Identity Authentication and click *Identity Authentication Admin Console*). You can manage the registered tenant for Identity Authentication as any other registered identity provider.

> **i Note**
>
> It will take about 2 minutes for the trust configuration with the tenant for Identity Authentication to become active.

> **i Note**
>
> Each SAP BTP subaccount is a separate service provider in the tenant for Identity Authentication .

> **→ Tip**
>
> If you need each of your SAP BTP applications to be represented by its own service provider, you must create and use a separate subaccount for each application. See Create a Subaccount.

Related Information

Identity Authentication
Application Identity Provider [page 1720]

7.1.1.3.4 On-Premise User Store

If you already have an existing on-premise system with a populated user store, you can configure SAP BTP applications to use that on-premise user store. This approach is similar to implementing identity federation with a corporate identity provider. In that way, applications do not need to keep the whole user database, but request the necessary information from the on-premise system.

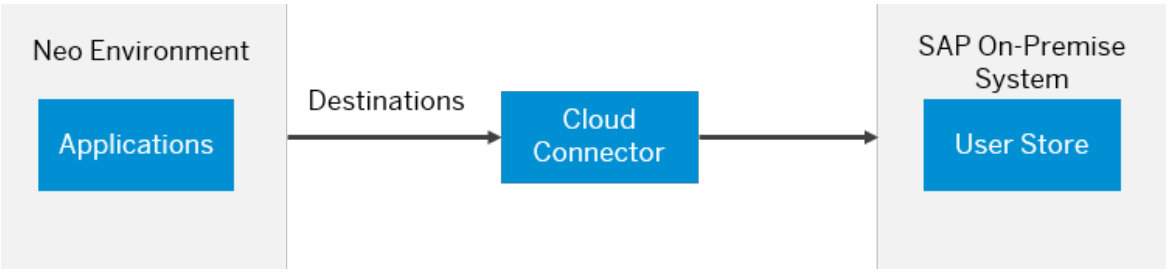
Context

Applications can use the on-premise system to:

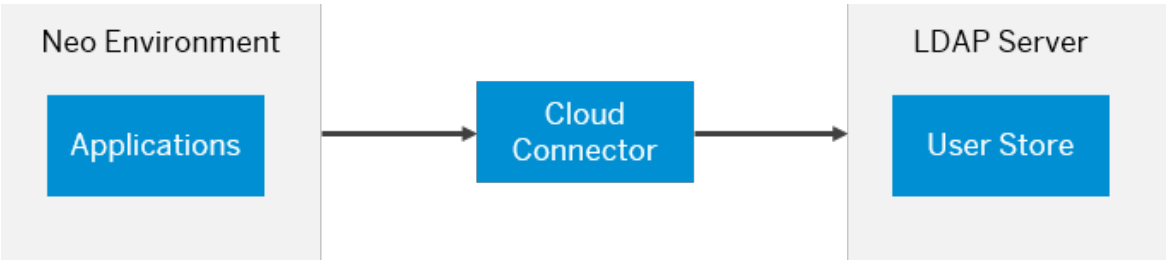
- check credentials
- search for users
- retrieve user details
- retrieve information about the groups a specific user is a member of. You can use this information for user authorizations. See Managing Roles [page 1710].

You can use two types of on-premise user store:

- SAP Single Sign-On with a SAP NetWeaver Application Server for Java System - the applications on SAP BTP connect to the SAP on-premise system using Destination API (and, if necessary, SAP HANA Cloud Connector), and make use of the user store there.



- Microsoft Active Directory - this is an LDAP server that can serve as an on-premise user store. The applications on SAP BTP connect to the LDAP server using SAP HANA cloud connector, and make use of the user store there.



Alternatively to the above scenarios, you can implement identity federation with a Identity Authentication tenant, where the tenant is configured to use an on-premise user store. See:

- Identity Authentication Tenant as an Application Identity Provider [page 1734]
- (Identity Authentication documentation) Corporate User Store

Related Information

Using an SAP System as an On-Premise User Store [page 1737]
Using Microsoft Active Directory as an On-Premise User Store [page 1740]

7.1.1.3.4.1 Using an SAP System as an On-Premise User Store

Overview

You can configure applications running on SAP BTP to use a user store of an SAP NetWeaver (7.2 or higher) Application Server for Java system and a SAP Single Sign-On system. That way SAP BTP does not need to keep the whole user database, but requests the necessary information from an on-premise system.

Prerequisites

- You have installed the SDK.
- You have set up the SDK location and landscape host.
- You have set up the console client.
  For more information, see Install the SAP BTP SDK for Neo Environment [page 845].
- You have a SAP NetWeaver 7.2 or higher Application Server for Java system
- You have installed and deployed federation software component archive (SCA) from SAP Single Sign-On (SSO) 2.0.
  For more information, see Downloading and Installing the Federation Software.

Context

1. Deploy the Application

When deploying the application, you have to set system properties of the application VM. For more information, see Configure VM Arguments [page 1598].

The properties are the following:

System Property	Value	Description
`com.sap.cloud.security.um.user_provider_name`	*onpremise*	This property specifies what user provider the application VM uses.
`com.sap.cloud.security.um.destination_name`	*<on-premise_destination_name>*	This property specifies the destination used by the on-premise user provider for the connection to the on-premise system. For more information about the destination, see Managing Destinations [page 54].

> **i Note**
>
> The WAR file that you are using as a source during the deployment has to be protected declaratively or programmatically. For more information, see Authentication [page 1678].

> **Example**
>
> ```
> neo deploy --host hana.ondemand.com --account myacc --application myapp --
> source samples/deploy_war/example.war --user mymail@example.com
> --vm-arguments "-Dcom.sap.cloud.security.um.user_provider_name=onpremise -
> Dcom.sap.cloud.security.um.destination_name=mydestination"
> ```

> **i Note**
>
> The VM arguments passed using this command will have effect only until you re-deploy the application.

2. Configure the On-Premise System

Context

The on-premise system is an AS Java with a deployed SCA from SAP Single Sign-On (SSO) 2.0. For the configuration of the on-premise AS Java system, proceed as follows:

Procedure

1. Configure a service user with SCIM_READONLY role.

   For more information about the role assignment process, see Assigning Principals to Roles or Groups.
2. If necessary, set the policy configuration to use the appropriate authentication method.

   You can use the following methods of authentication:

   - Basic authentication

The on-premise AS Java system is configured to use basic authentication by default. That means the `sap.com/tc~sec~scim~server*scim_v1` policy configuration is set to use *basic* Web template by default.

- Client certificate authentication
  For client certificate authentication, you need to update the `sap.com/tc~sec~scim~server*scim_v1` policy configuration to use *client_cert* Web template. In addition, you have to configure the on-premise system to use the client certificate properly. For more information about the on-premise system configuration, see Using X.509 Client Certificates on the AS Java.

For more information about the policy configuration, see Editing the Authentication Policy of AS Java Components.

3. If your user does not exist in the on-premise system, create a technical user.

   For more information, see Creating a Technical User.

3. Configure the Destination to the On-Premise System

For the proper communication with the on-premise AS Java system, you need to configure the destination of the Java application on SAP BTP. For more information, see Configure Destinations from the Cockpit [page 77].

You have to set the following properties for the destination of the cloud application:

Destination Property	Value	Description
Name	*<on-premise_destination_name>*	The name of the destination must match with the value of system property `com.sap.cloud.security.um.destination_name`.
Type	*HTTP*	For more information, see HTTP Destinations [page 91].
URL	*https:// < AS Java Host>:<AS Java HTTPS Port>/scim/v1/* Or *http://<Virtual host configured in Cloud Connector>:<virtual Port>/scim/v1/*	The URL to the on-premise AS Java system if it is exposed via reverse proxy. Or in case the on-premise systems is exposed via HANA Cloud Connector the virtual URL configured in Cloud Connector. In this case, the configured protocol should be http as the connectivity service is using secure tunneling to the on-premise system.

Destination Property	Value	Description
Proxy Type	*Internet* or *OnPremise*	• Internet If you use an internet proxy, you have to make sure that the on-premise system is accessible from the application VM. • OnPremise If you use an on-premise proxy, you have to install SAP HANA cloud connector and configure a tunnel from the destination URL to the on-premise system. For more information about SAP HANA cloud connector, see Cloud Connector.
Authentication	*BasicAuthentication* or *ClientCertificateAuthentication*	For the configuration of such an authentication, you need to specify the credentials of the service user from the on-premise system. For more information about the destination configuration, see HTTP Destinations [page 91].
User	*<user name>*	This property is used for basic authentication only, and it specifies the name of the service user in the on-premise AS Java system.
Password	*<password>*	This property is used for basic authentication only, and it specifies the service user's password.

7.1.1.3.4.2 Using Microsoft Active Directory as an On-Premise User Store

You can use Microsoft Active Directory as an on-premise LDAP server providing a user store for your SAP BTP applications.

Prerequisites

- You have installed the SDK.
- You have set up the SDK location and landscape host.
- You have set up the console client.
  For more information, see Install the SAP BTP SDK for Neo Environment [page 845].
- You have installed and deployed SAP HANA cloud connector. See Installation [page 238].

Context

1. Deploy the Application

When deploying the application, you have to set system properties of the application VM. For more information, see Configure VM Arguments [page 1598].

The properties are the following:

System Property	Value	Description
`com.sap.cloud.security.um.user_provider_name`	*onpremise*	This property specifies what user provider the application VM uses.
`com.sap.cloud.security.um.destination_name`	*<on-premise_destination_name>*	This property specifies the destination used by the on-premise user provider for the connection to the on-premise system. For more information about the destination, see Managing Destinations [page 54].

> **i Note**
>
> The WAR file that you are using as a source during the deployment has to be protected declaratively or programmatically. For more information, see Authentication [page 1678].

> **Example**
>
> ```
> neo deploy --host hana.ondemand.com --account mysubaccount --application
> myapp --source samples/deploy_war/example.war --user mymail@example.com
> --vm-arguments "-Dcom.sap.cloud.security.um.user_provider_name=onpremise -
> Dcom.sap.cloud.security.um.destination_name=mydestination"
> ```

> **i Note**
>
> The VM arguments passed using this command will have effect only until you re-deploy the application.

2. Install and Configure SAP HANA Cloud Connector

Create the required destination and configure SAP HANA clolud connector as described in Configure an On-Premise User Store [page 441].

7.1.1.3.5 Authentication Configuration

This is an optional procedure that you can perform to configure the authentication methods used in a cloud application. You can configure the behavior of standard Java EE authentication methods, or define custom ones, based on custom combinations of login options.

Prerequisites

- You have an application with authentication defined in its `web.xml` or source code. See Authentication [page 1678].

Context

The following table describes the available login options. In the default authentication configuration, they are pre-assigned to standard Java EE authentication methods. If you want to change this, you need to create a custom configuration.

For each authentication method, you can select a custom combination of options. You may need to select more than one option if you want to enable more than one way for users to authenticate for this application.

If you select more than one option, SAP BTP will delegate authentication to the relevant login modules consecutively in a stack. When a login module succeeds to authenticate the user, authentication ends with success. If no login module succeeds, authentication fails.

Login Option	Descrption
Trusted SAML 2.0 identity provider	Authentication is implemented over the Security Assertion Markup Language (SAML) 2.0 protocol, and delegated to SAP ID service or custom identity provider (IdP). The credentials users need to present depend on the IdP settings. See Application Identity Provider [page 1720].
User name and password	HTTP BASIC authentication with user name and password. The user name and password are validated either by SAP ID service (default) or by an on-premise SAP NetWeaver AS Java. See Using an SAP System as an On-Premise User Store [page 1737]. **i Note** If you want to use your Identity Authentication tenant for BASIC authentication (instead of SAP ID service/SAP NetWeaver), create a customer ticket in component **BC-NEO-SEC-IAM**. In the ticket, specify the Identity Authentication tenant you want to use.
Client certificate	Users authenticate with a client certificate installed in an on-premise SAP NetWeaver Application Server for Java system. See Enabling Client Certificate Authentication [page 1797]

Login Option	Descrption
Application-to-Application SSO	Used for AppToAppSSO destinations. See Application-to-Application SSO Authentication [page 104]. **i Note** When you select *Trusted SAML 2.0 identity provider*, *Application-to-Application SSO* becomes enabled automatically.
OAuth 2.0 token	Authentication is implemented over the OAuth 2.0 protocol. Users need to present an OAuth access token as credential. See OAuth 2.0 Authorization Code Grant [page 1753].

Procedure

1. In the SAP BTP cockpit, navigate to the required SAP BTP subaccount. See Navigate in the Cockpit.
2. Enter the *Applications* > *Java Applications* section.
3. Click the application you want to configure.
4. Enter the *Authentication Configuration* section.
5. To configure the default settings, choose *Activate Custom Configuration*.

   You can configure existing authentication methods or create new ones. If you need to restore the default state of all default methods, choose the *Reset Custom Configuration* button for the entire panel. If you need to restore the default state of a particular method, choose the *Reset Authentication Method* button for that method (not available for custom methods you defined).
6. Save the changes to the authentication configuration.
7. Restart the application so the changes can take effect.

Example

You have a Web application that users access using a Web browser. You want users to log in using a SAML identity provider. Hence, you define the FORM authentication method in the `web.xml` of the application. However, later you decide to provide mobile access to your application using the OAuth protocol (SAML is not optimized for mobile access). You do this by adding the *OAuth 2.0 token* option for the FORM method for your application. In this way, desktop users will continue to log in using a SAML identity provider, and mobile users will use an OAuth 2.0 access token.

Related Information

Authentication [page 1678]

Specifying Authentication Mechanisms (general information)

7.1.2 Securing HTML5 Applications

The security guide provides an overview of the security-relevant information that applies to HTML5 applications.

Related Information

Authentication [page 1744]
Authorization [page 1744]
Accessing On-Premise Systems [page 1745]
Protecting Applications from Cross-Site Scripting (XSS) [page 1745]
Protecting from Cross-Site Request Forgery (CSRF) [page 1745]

7.1.2.1 Authentication

SAP BTP uses the Security Assertion Markup Language (SAML) 2.0 protocol for authentication and single sign-on.

By default, the SAP BTP is configured to use the SAP ID service as identity provider (IdP), as specified in SAML 2.0. You can configure a trust relationship to your custom IdP to provide access to the cloud using your own user database. For information, see Application Identity Provider [page 1720].

HTML5 applications are protected with SAML2 authentication by default. For publicly accessible applications, the authentication can be switched off. For information about how to switch off authentication, see Authentication [page 1130].

7.1.2.2 Authorization

HTML5 applications may be protected by permissions.

Permissions for an HTML5 application are defined in the application descriptor file. For more information about how to define permissions for an HTML5 application, see Authorization [page 1131].

Permissions defined in the application descriptor are only effective for the active application version. To protect non-active application versions, the default permission `NonActiveApplicationPermission` is defined by the system for every HTML5 application.

To assign users to a permission of an HTML5 application, a role must be assigned to the corresponding permission. As a result, all users who are assigned to the role get the corresponding permission. Roles are not application-specific but can be reused across multiple HTML5 applications. For more information about creating roles and assigning roles to permissions, see Managing Roles and Permissions [page 1637].

> **i Note**
>
> HTML5 application permissions can only protect the access to the REST service through the HTML5 application. If the REST service is otherwise accessible on the Internet or a corporate network, it must implement its own authentication and authorization concept..

7.1.2.3 Accessing On-Premise Systems

To access a system that is running in an on-premise network, you can set up an SSL tunnel from your on-premise network to the SAP BTP using the SAP BTP Cloud Connector.

For more information about setting up the Cloud connector, see the *Cloud Connector Operator's Guide*.

Related Information

7.1.2.4 Protecting Applications from Cross-Site Scripting (XSS)

Cross-site scripting (XSS) is one of the most common types of malicious attacks on web applications.

If an HTML5 application is connected to a REST service, the corresponding REST service must take measures to protect the application against this type of vulnerabilities. For REST services implemented on the SAP BTP a common output encoding library may be used to protect applications. For more information about XSS protection on the SAP BTP, see Protection from Cross-Site Scripting (XSS) [page 1836].

7.1.2.5 Protecting from Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) is another common type of attack on web applications.

If an application connects to a REST service, the corresponding REST service must take measures to protect against CSRF. For REST services implemented on the SAP BTP a CSRF prevention filter may be used in the corresponding REST service. For more information about CSRF protection on the SAP BTP, see Protection from Cross-Site Request Forgery [page 1840].

Related Information

Application Identity Provider [page 1720]

Test Security on the Cloud (with a Local Identity Provider) [page 1703]

7.1.3 Securing SAP HANA Applications

In this section, you can find information relevant for securing SAP HANA applications running on SAP BTP.

Security Information

Info Type	See
General security concepts for SAP HANA applications	SAP HANA Security Guide
Specific security concepts for SAP HANA applications running on SAP BTP	Configure SAML 2.0 Authentication [page 1015]

7.2 Platform Identity Provider

The platform identity provider is the user base for access to your SAP BTP subaccount in the Neo environment. The default user base is provided by SAP ID Service. You can switch to an Identity Authentication tenant if you want to use a custom user base.

Task overview: Security, Neo Environment [page 1671]

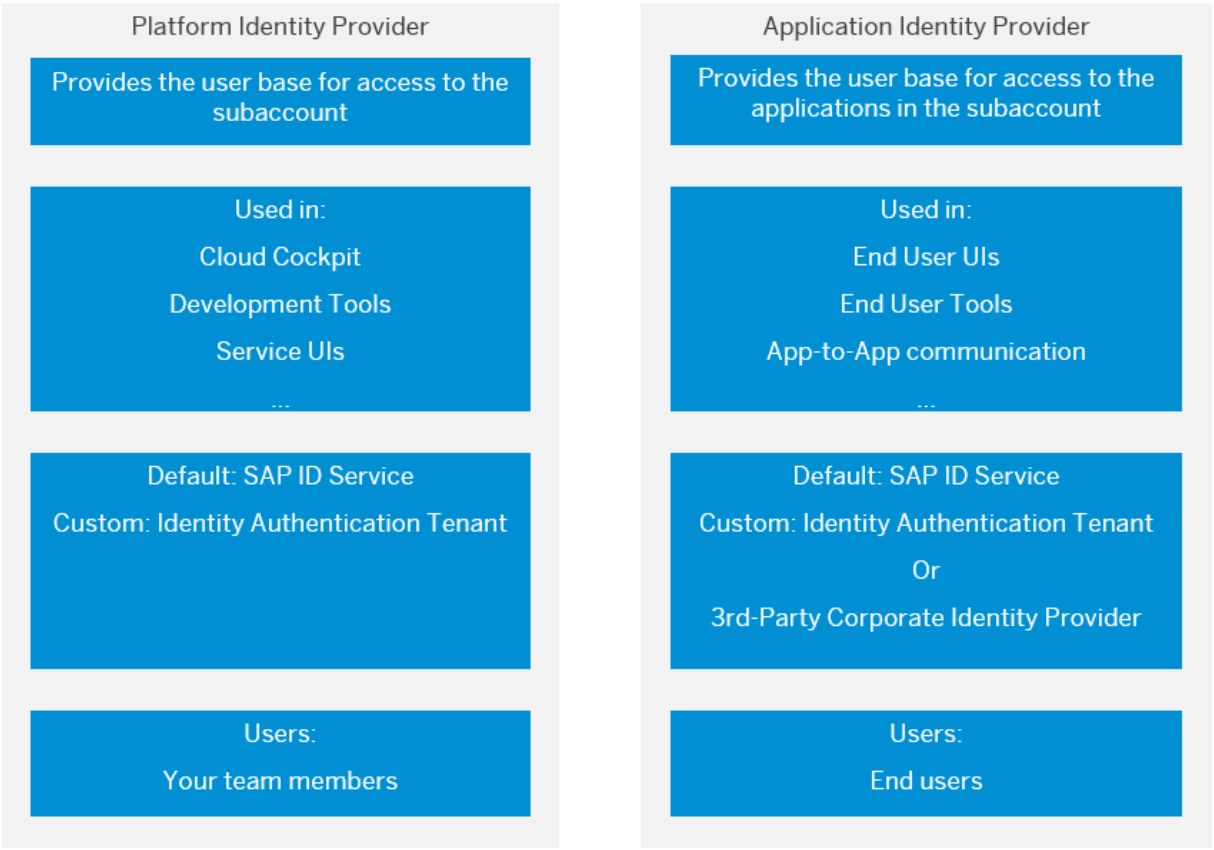
Related Information

Authorization and Trust Management in the Neo Environment [page 1672]
OAuth 2.0 Service [page 1751]
Keystore Service [page 1779]
Audit Logging in the Neo Environment [page 1810]
Principal Propagation [page 1816]
Protection from Web Attacks [page 1835]
Data Protection and Privacy [page 1848]

Overview

By default, the SAP BTP cockpit and console client are configured to use SAP ID Service as the **platform identity provider** (providing the user base for subaccount members). SAP ID Service, however, uses the SAP user base (providing, for example, your s- or p-user). If you want to have subaccount members from your custom user base, and use custom security configuration (such as two-factor user authentication, or corporate user store, for example), you can switch to a custom Identity Authentication tenant as a platform identity provider.

> **i Note**
>
> There is a difference between a **platform** identity provider and **application** identity provider at SAP BTP.
>
> The diagram below describes the basic features of platform identity providers and application identity providers, and provides a brief comparison between them.



> **i Note**
>
> Changing the **platform** identity provider settings (▶ *Security* ▶ *Trust* ▶ *Platform Identity Provider* ◀ in the SAP BTP cockpit) does **not** affect the **application** identity provider settings (▶ *Security* ▶ *Trust* ▶ *Platform Identity Provider* ◀ in the SAP BTP cockpit) for this subaccount. See Application Identity Provider [page 1720].

1. Create Trust with the Identity Authentication Tenant

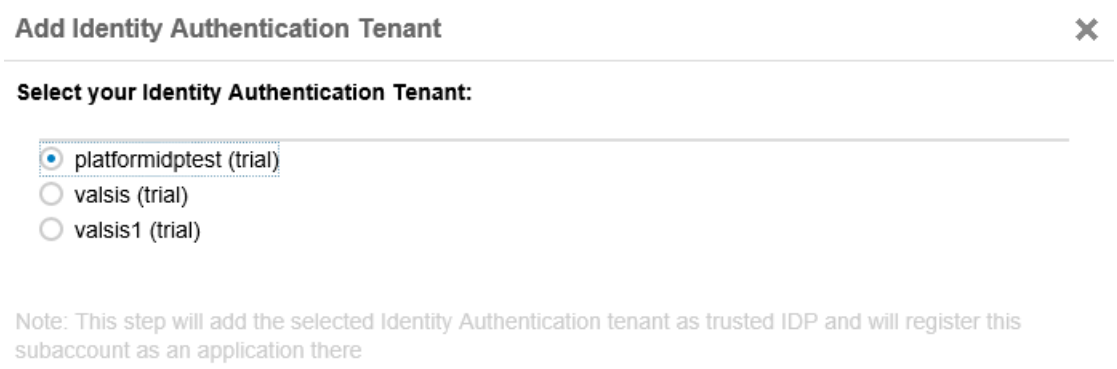
Prerequisites

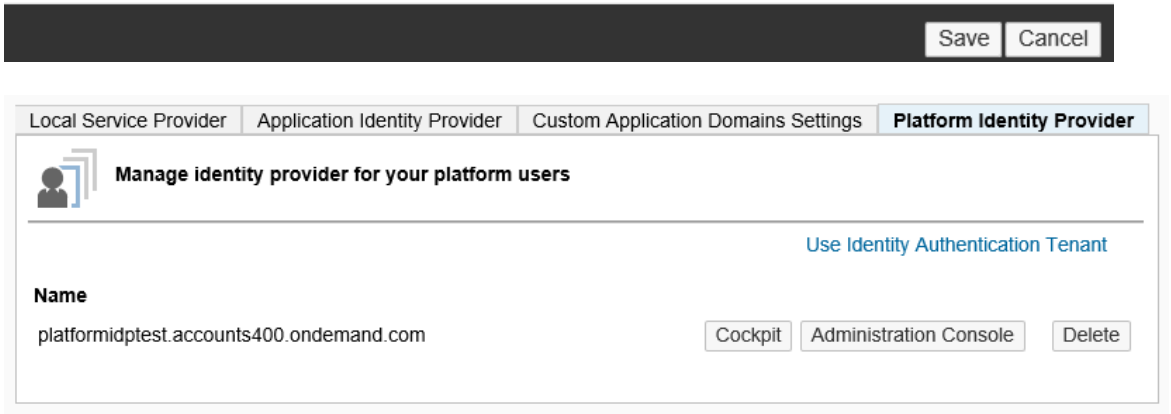
- You have a user with Administrator role for your subaccount (provided by the default user base, SAP ID Service).
- You have enabled the Platform Identity Provider service. See Using Services in the Neo Environment [page 1152].

- You have an Identity Authentication tenant configured. See Identity Authentication documentation.

Procedure

1. Log in to the SAP BTP cockpit with the Administrator user from the default user base.
2. Navigate to the required SAP BTP subaccount. See Navigate in the Cockpit.
3. Go to the *Security* > *Trust* section.
4. Go to the *Platform Identity Provider* tab.
5. Choose *Use Identity Authentication Tenant*.
6. Choose the required tenant and save.





The Identity Authentication tenant appears as a platform identity provider. The trust configuration with it is complete. You can proceed with adding tenant users as subaccount members, and the rest of the steps described in this document.

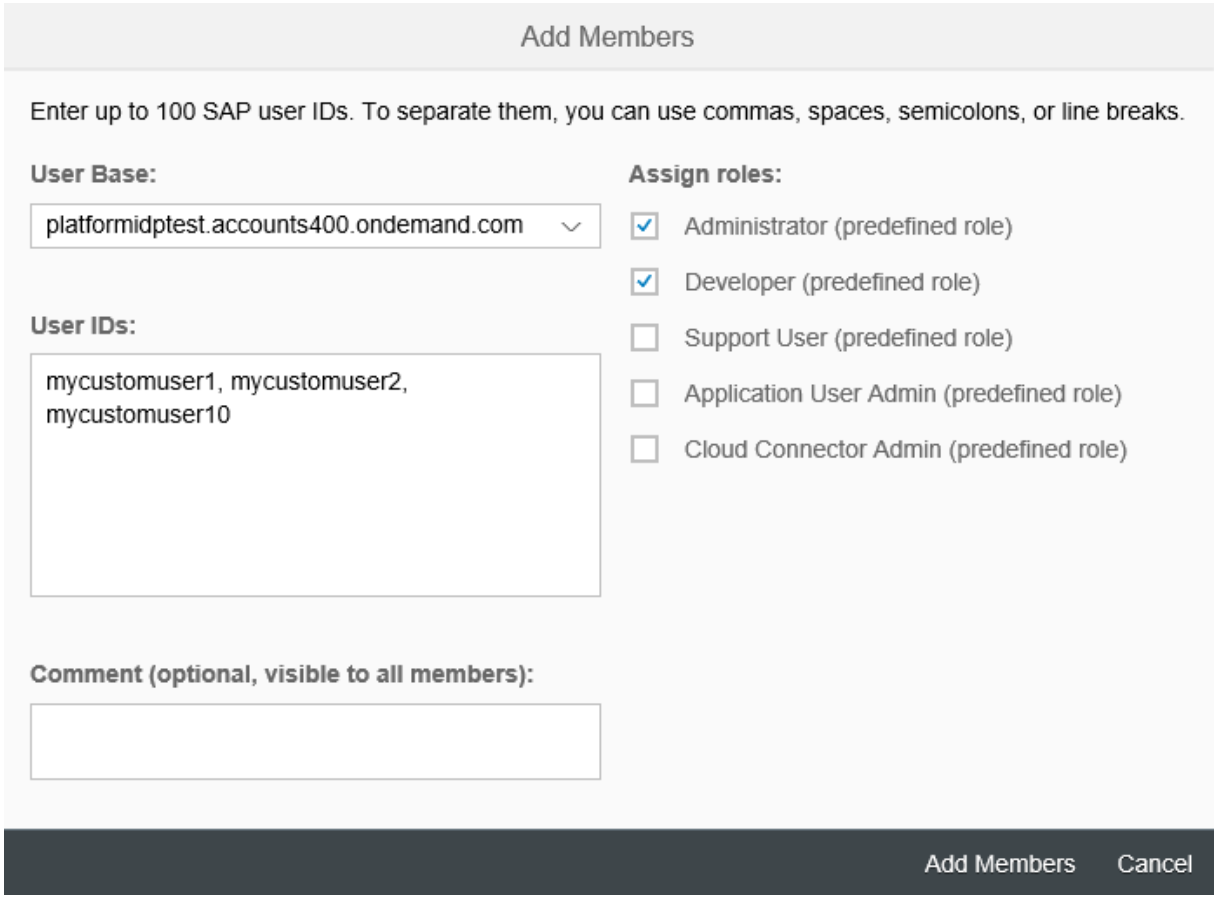
2. Add Identity Authentication Tenant Users as Subaccount Members

Context

Now that you have switched the user base, you need to add the users that you will use for access to this subaccount as subaccount members.

Go to the *Members* tab in the SAP BTP cockpit. You can see all cockpit users, with their IDs, roles and user base, listed here. To add a new member, choose *Add Members* and configure the member users from the respective user base (Identity Authentication tenant). See also Add Members to Your Neo Subaccount [page 1299].

> **i Note**
>
> The account members for access to this subaccount from the console client must have Administrator role.



(Optional) 3. Configure the Identity Authentication Tenant for the Required Scenarios

Context

You can configure the Identity Authentication tenant for specific authentication scenarios using its Administration Console UI.

To do so, choose the *Administration Console* button next to the registered tenant in the *Security* > *Trust* > *Platform Identity Provider* section of the SAP BTP cockpit.

In the tenant's Administration Console you will notice it displays the SAP BTP cockpit as a registered application. The application has `<Identity Authentication tenant ID>` as *display name*, and `https://account.hana.ondemand.com/<account name>/admin` as *SP name*.

Related information in the documentation of Identity Authentication:

- Corporate User Store
- Configure Kerberos Authentication
- User Management
- Configure Risk-Based Authentication
- Operation Guide

Accessing the SAP BTP Cockpit with the Tenant User Base

Context

If you open the default cockpit URL, `https://account.<SAP BTP host>/cockpit`, SAP ID Service will be used for user authentication.

To request the SAP BTP cockpit using the Identity Authentication tenant user base, use the following URL:

`https://account-<subaccount-name>.<SAP BTP host>`

For the SAP BTP host, see Regions.

> → Tip
>
> Make sure you use the **subaccount name**, not the subaccount **display** name, which could be different. Check the value of the subaccount name in the subaccount overview section in the cloud cockpit.

> ℹ Note
>
> - You can see only those subaccounts that are in the region of the tenant cockpit URL.
> - If you want to use risk-based authentication, for example, to enable two-factor authentication (TFA), you have to enable it for **all** subaccounts in your global account. This means for each subaccount you need to configure the platform identity provider to be an Identity Authentication tenant configured properly for risk-based authentication.

> For more information about TFA in your Identity Authentication tenant, see Two-Factor Authentication.

Procedure

1. In an incognito browser window, open the tenant cockpit URL. This is required to make sure you are not logged in with the SAP ID Service user.
2. Log in with a user name and password from the Identity Authentication tenant.

Using the Console Client with the Tenant User Base

Context

When using the console client with a custom platform identity provider, you must supply a user from your custom Identity Authentication tenant. For example, you want to execute the `list-schemas` command. In the corresponding command parameter, you can provide the login id or email address of your user in the Identity Authentication tenant as follows:

```
neo list-schemas -a <name of the subaccount> -h <region host> -u <login id or
email of the user>
```

If you have enabled two-factor authentication (TFA) in your Identity Authentication tenant, you can enter the 6-digit passcode after the user's password when the console client prompts you for password.

For more information about two-factor authentication in your Identity Authentication tenant, see Two-Factor Authentication.

> → Tip
>
> If you want to switch back to the default user base of SAP ID Service in the console client, you need to remove the custom platform identity provider configuration you created.

7.3 OAuth 2.0 Service

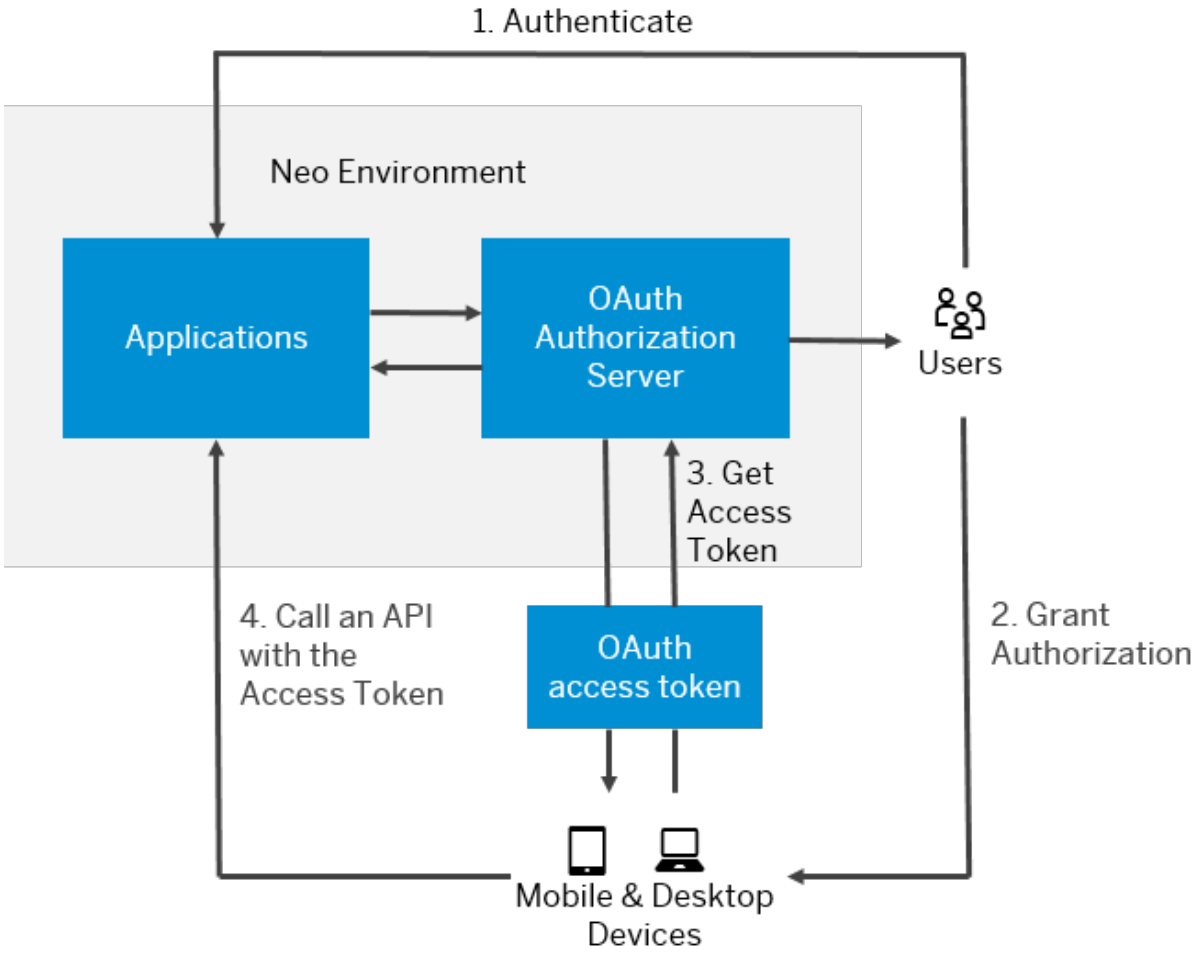
Use OAuth 2.0 service on SAP BTP to protect applications in the Neo environment using the OAuth 2.0 protocol.

Protecting Applications with OAuth 2.0

OAuth 2.0 is a widely adopted security protocol for protection of resources over the Internet. It is used by many social network providers and by corporate networks. It allows an application to request authentication on

behalf of users with third-party user accounts, without the user having to grant its credentials to the application. SAP BTP provides an API for developing OAuth-protected applications. You can configure the required scopes and clients using the SAP BTP cockpit.

The following graphic illustrates protecting applications with OAuth on SAP BTP.



SAP BTP supports two basic OAuth 2.0 flows:

- Authorization code grant - there is a human user who authorizes a mobile application to access resources on his or her behalf. See OAuth 2.0 Authorization Code Grant [page 1753].
- Client credentials grant - there is no human user but a device instead. In such case, the access token is granted on the basis of client credentials only. See OAuth 2.0 Client Credentials Grant [page 1759].

Parent topic: Security, Neo Environment [page 1671]

Related Information

Authorization and Trust Management in the Neo Environment [page 1672]
Platform Identity Provider [page 1746]

Keystore Service [page 1779]
Audit Logging in the Neo Environment [page 1810]
Principal Propagation [page 1816]
Protection from Web Attacks [page 1835]
Data Protection and Privacy [page 1848]
OAuth 2.0 Authorization Code Grant [page 1753]
OAuth 2.0 Client Credentials Grant [page 1759]
OAuth 2.0 Configuration [page 1760]
Principal Propagation to OAuth-Protected Applications [page 1770]

7.3.1 OAuth 2.0 Authorization Code Grant

Use OAuth 2.0 service in the Neo environment of SAP BTP to enable your cloud applications for authorization code grant flow. Authorization code grant is one of the basic flows specified in the OAuth 2.0 protocol.

Overview

OAuth 2.0

OAuth has taken off as a standard way and a best practice for applications and websites to handle authorization. OAuth defines an open protocol for allowing secure API authorization of desktop, mobile and web applications through a simple and standard method.

OAuth is based on granting access without explicit credentials sharing. OAuth:

- Avoids storing credentials at the third-party location
- Limits the access permissions granted to third parties
- Enables easy access right revocation without the need to change credentials

In this way, OAuth mitigates some of the common concerns with authorization scenarios.

The following table shows the roles defined by OAuth, and their respective entities in SAP BTP:

Role	Entity in SAP BTP	Description
Resource owner	User	An entity that holds protected assets. This entity is capable of granting access to those assets under its control.
Resource server	Application	The server that hosts the resource owner's protected assets.
Client	Third-party application	The third party entity that needs to access the protected assets on behalf of the resource owner.

Role	Entity in SAP BTP	Description
Authorization server	SAP BTP infrastructure	The server that manages the authentication and authorization of the different entities involved.

For more information, see the OAuth 2.0 Specification .

Using OAuth in SAP BTP

Protecting Resources Declaratively

If you want to implement a login based on credentials in the form of an OAuth token, you can do that by using OAuth as a login method in your application web.xml. For example:

```
<login-config>
  <auth-method>OAUTH</auth-method>
</login-config>
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Protected Area</web-resource-name>
    <url-pattern>/rest/get-photos</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <!-- Role Everyone will not be assignable -->
    <role-name>Everyone</role-name>
    </auth-constraint>
</security-constraint>
<security-role>
  <description>All SAP BTP users</description>
  <role-name>Everyone</role-name>
</security-role>
```

For more information, see Enabling Authentication [page 1678]

Working with User Attributes

In your protected application you can acquire the user ID and attributes as described in Working with User Profile Attributes [page 1696].

There are two additional user attributes you can use to retrieve token specific information:

- `com.sap.security.oauth2.clientId` - holds information about the OAuth client ID
- `com.sap.security.oauth2.grantedScopes` - holds information about the granted scopes.

Handling Sessions

The Java EE specification requires session support on the client side. Sessions are maintained with a cookie which the client receives during the authentication and then passes it along to the server on every request. The OAuth specification, however, does not necessarily require the client to support such a session mechanism. That is, the support of cookies is not mandatory. On every request, the client passes along to the server only the token instead of passing cookies. Using the OAuth login module described in the Protecting Resources Declaratively section, you can implement a user login based on an access token. The login, however, occurs on every request, and thus it implies the risk of creating too many sessions in the Web container.

To use requests that do not hold a Web container session, use a filter with the proper configuration, as described in the following example:

```
<filter>
    <display-name>OAuth scope definition for viewing a photo album</display-name>
    <filter-name>OAuthViewPhotosScopeFilter</filter-name>
    <filter-class>
      com.sap.cloud.security.oauth2.OAuthAuthorizationFilter
    </filter-class>
  <init-param>
    <param-name>scope</param-name>
    <param-value>view-photos_upload-photos</param-value>
  </init-param>
      <init-param>
    <param-name>no-session</param-name>
    <param-value>false</param-value>
  </init-param>
</filter>
```

Checking Scopes Declaratively

One of the ways to enforce scope checks for resources is to declare the resource protection in the web.xml. This is done by specifying the following elements:

Element	Description
Servlet filter class	Enter as value **`com.sap.cloud.security.oauth2.OAuthAuthorizationFilter`**. See OAuthAuthorizationFilter. On request it checks if the request contains a valid OAuth token to access the resources mapped to the configured scope.
Protected resources	Could be given as URL pattern or servlet.
Initial parameters	With these, you specify the scope, user principal and HTTP method: • `scope` • `http-method` • `user-principal` - if set to "yes", you will get the user ID • `no-session` - if you set this to "true", the session will be destroyed when you finish using the filter. This means that each time the filter is used, a new session will be created. Default value: false.

The following example shows a sample web.xml for defining and configuring OAuth resource protection for the application.

```
<filter>
    <display-name>OAuth scope definition for viewing a photo album</display-name>
    <filter-name>OAuthViewPhotosScopeFilter</filter-name>
    <filter-class>
      com.sap.cloud.security.oauth2.OAuthAuthorizationFilter
    </filter-class>
  <init-param>
    <param-name>scope</param-name>
    <param-value>view-photos</param-value>
  </init-param>
```

```
  <init-param>
    <param-name>http-method</param-name>
    <param-value>get post</param-value>
  </init-param>
</filter>
```

Declaring resource-to-scope Filter Mapping

In this code snippet you can observe how the `PhotoAlbumServlet` is mapped to the previously specified OAuth scope filter:

```
<filter-mapping>
  <filter-name>OAuthViewPhotosScopeFilter</filter-name>
  <servlet-name>PhotoAlbumServlet</servlet-name>
</filter-mapping>
```

If you would like to use URL pattern instead, simply specify the pattern that should apply here:

```
<filter-mapping>
  <filter-name>OAuthViewPhotosScopeFilter</filter-name>
  <url-pattern>/photos/*.jpg</url-pattern>
</filter-mapping>
```

In the second case, all files with the `*.jpg` extension that are served from the `/photos` directory will be protected by the OAuth filter.

For more information regarding possible mappings, see the `filter-mapping` element specification.

Protecting Resources Programmatically

Alternatively to the declarative approach with the `web.xml` (described above), you can use the OAUTH login module programmatically. For more information, see Programmatic Authentication [page 1683].

Creating Protected Resource Requests

When a resource protected by OAuth is requested, your application must pass the access token using the HTTP "Authorization" request header field. The value of this header must be the token type and access token value. The currently supported token type is "bearer".

Possible Responses to Requests for Protected Resources

When the protected resource access check is performed the filter calls the API and the API calls the authorization server to check the validity of the access token and retrieve token's scopes.

In the table below the result handling between the authorization server and resource server, resource server and the API, and resource server and filter is presented.

Responses to requests for protected resources

Authorization server to resource server		Resource server to the API		Resource server to the filter	
Code	**Description**	**Return value / Exception**	**Description**	**Code**	**Description**
`200`	`access_token` is valid	`True`	attribute "`user_id`" in the request attribute "`client_id`" in the request		Access is allowed. attribute "`client_id`" in the request attribute "`user_id`" in the request If `user-principal=true` -> `request.getUserPrincipal().getName()` returns `user_id`
`200`	`access_token` is valid	`False`	attribute "`reason`" in the request describing the reason for the result `reason = "access_forbidden"`	`403`	Access is forbidden
`400`	`access_token` parameter is null, empty string, missing or it is given more than once	`False`	Attribute "`reason`" in the request describing the reason for the result `reason = "missing_access_token`	`401`	
`401`	`access_token` does not exist	`False`	Attribute "`reason`" in the request describing the reason for the result `reason = "missing_access_token`	`401`	

Authorization server to resource server		Resource server to the API		Resource server to the filter
`401`	`access_token` has expired	`False`	Attribute `"reason"` in the request describing the reason for the result `reason = "missing_access_token`	`401`
`401`	`access_token` is not issued for the current subscription	`False`	Attribute `"reason"` in the request describing the reason for the result `reason = "missing_access_token`	`401`
`500`	Unexpected error (no connection to the database)	`OAuthSystemException` (extends `Exception`)	Inherit message from the original exception	`500`
		`OAuthSystemException` (extends `Exception`)	HTTP request to the authorization server fails	
		`OAuthSystemException` (extends `Exception`)	OAuth destination is not found or can't get destination HTTP client	

Next Steps

1. You can now deploy the application on SAP BTP. For more information, see Deploying and Updating Applications [page 875]
2. After you deploy, you need to configure clients and scopes for the application. For more information, see OAuth 2.0 Configuration [page 1760].

7.3.2 OAuth 2.0 Client Credentials Grant

Use OAuth 2.0 service in the Neo environment of SAP BTP to enable your cloud applications for client credentials grant flow.

Context

Client credentials grant is one of the basic flows specified in the OAuth 2.0 protocol. It enables grant of an OAuth access token based on the client credentials only, without user interaction. You can use this flow for enabling system-to-system communication (with a service user), for example, in device communication in an Internet of things scenario.

Procedure

1. Register a new OAuth client of type *Confidential*. See Register an OAuth Client [page 1760].
2. Using that client, you can get an access token using a REST call to the endpoints shown in *cockpit* > *Security* > *OAuth* > *Branding*.

   Create a REST call containing `grant_type: client_credentials`, client ID and password.

   See the OAuth 2.0 client credentials grant specification .

   The HTTP response contains the access token.
3. In the SAP BTP application:
   - Protect your application declaratively with the OAuth login method in the `web.xml`. See OAuth 2.0 Authorization Code Grant [page 1753].
   - Use the `getRemoteUser()` method of the HTTP request (`javax.servlet.http.HttpServletRequest`) to get the **client ID**.
     The `getRemoteUser()` method returns the client ID prefixed by `oauth_client_` as follows: `oauth_client_<client ID>`

     > → Tip
     >
     > You can use the client ID returned as remote user to assign Java EE roles to clients, and use them for role-based authorizations. See:
     >
     > - Managing Roles [page 1710]
     > - Authorizations [page 1686]

     > ⚠ Caution
     >
     > Having multiple clients with the same case-sensitive name will lead to having the same user ID at runtime. This could lead to incorrect user role assignments and authorizations.

7.3.3 OAuth 2.0 Configuration

Register clients, manage access tokens, configure scopes and perform other OAuth configuration tasks in the Neo environment of SAP BTP.

Prerequisites

- You have an account with administrator role in SAP BTP. See Managing Member Authorizations in the Neo Environment [page 1301].
- You have developed an OAuth-protected application (resource server). See OAuth 2.0 Authorization Code Grant [page 1753].
- You have deployed the application on SAP BTP. See Deploying and Updating Applications [page 875].

Contents:

- Register an OAuth Client [page 1760]
- Define OAuth Scopes [page 1763]
- Revoke OAuth Access Tokens [page 1764]
- Use a QR Code for Mobile Access [page 1765]
- Customize Corporate Branding [page 1767]

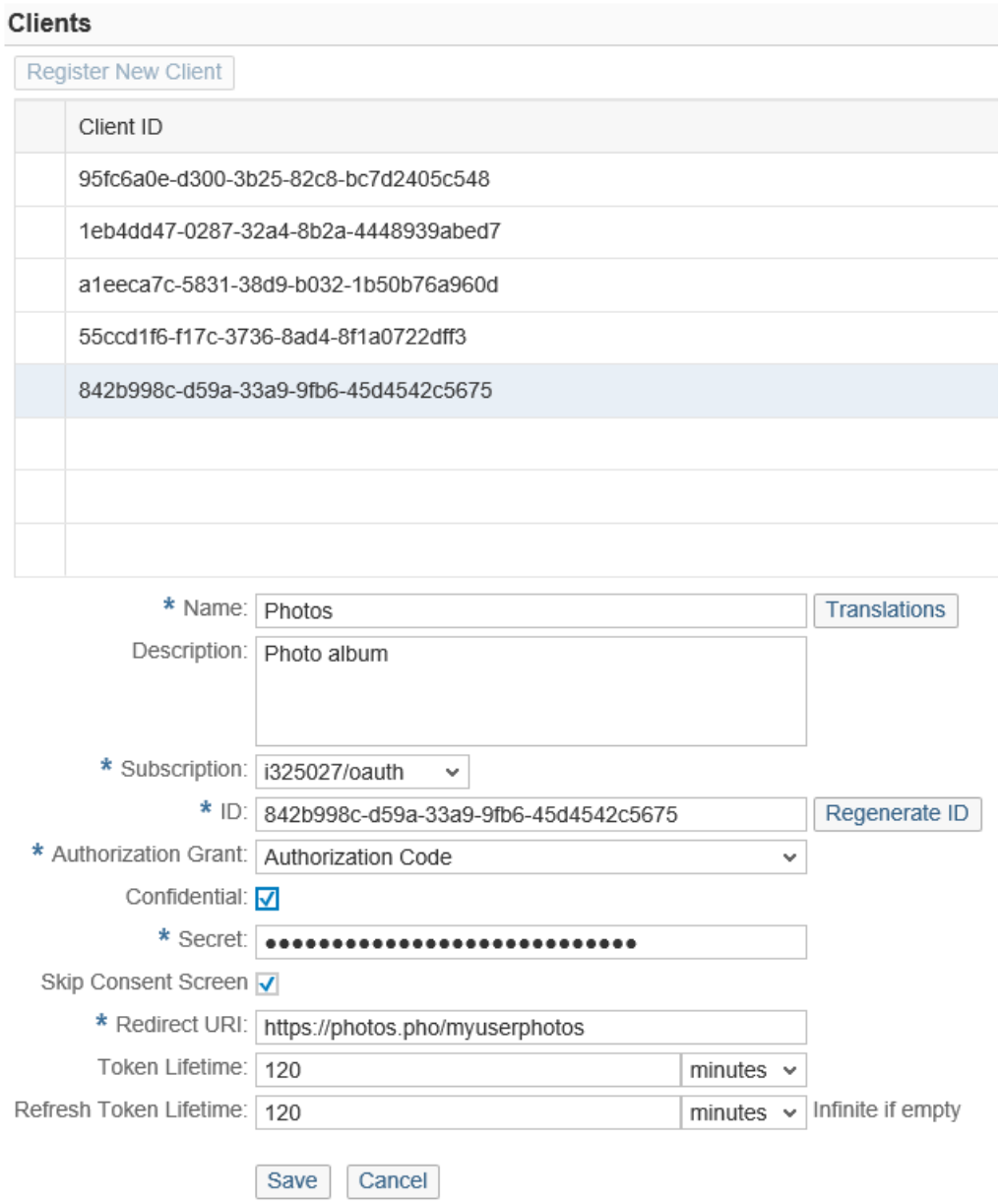
7.3.3.1 Register an OAuth Client

To authorize a device to access an OAuth-protected application, you need to register it as a client.

Context

Procedure

1. In your Web browser, log on to the cockpit, and select an account.
2. In the *Security* > *OAuth* section, go to the *Clients* tab.
3. Choose *Register new Client*.



4. Enter the client data as required (see the table below).

Field	Description
Name	The client name.
Description	A free-text description of the client.
Subscription	The application for which you are registering this client. To be able to register for a particular application, this account must be subscribed to it. For more information, see Register an OAuth Client [page 1760].
ID	Required. The ID of the client authorized to access the resource server running on SAP BTP. If you already have a client with a defined ID at the client device, enter its

Field	Description
	value here. Otherwise, you can choose *Generate ID* and *Secret* to use a system-generated ID, or enter a custom value. In that case, you must provide that value to the user of the client device. **i Note** The client ID must be globally unique within the entire SAP BTP.
Confidential	If you mark this box, the client ID will be protected with a password. You will need to supply the password here, and provide it to the client.
Secret	A secret (password) that allows the authorization server to authenticate before the client on behalf of the resource owner (user). It will also be needed by the client.
Skip Consent Screen	If you mark this option, no end user action will be required for authorizing this client. Otherwise, the end user will have to confirm granting the requested authorization.
Redirect URI	The application URI to which the authorization server will connect the client with the authorization code.
Token Lifetime	The token lifetime.This value applies to the access token and authorization code.
Refresh Token Lifetime	The refresh token lifetime.
Translations	Optionally, you can provide translations of the client name and description for localization purposes. Choose the *Translations* button and enter the required language translation there.

5. Save the new client.

Results

The device with the defined ID will be recognized as a registered client.

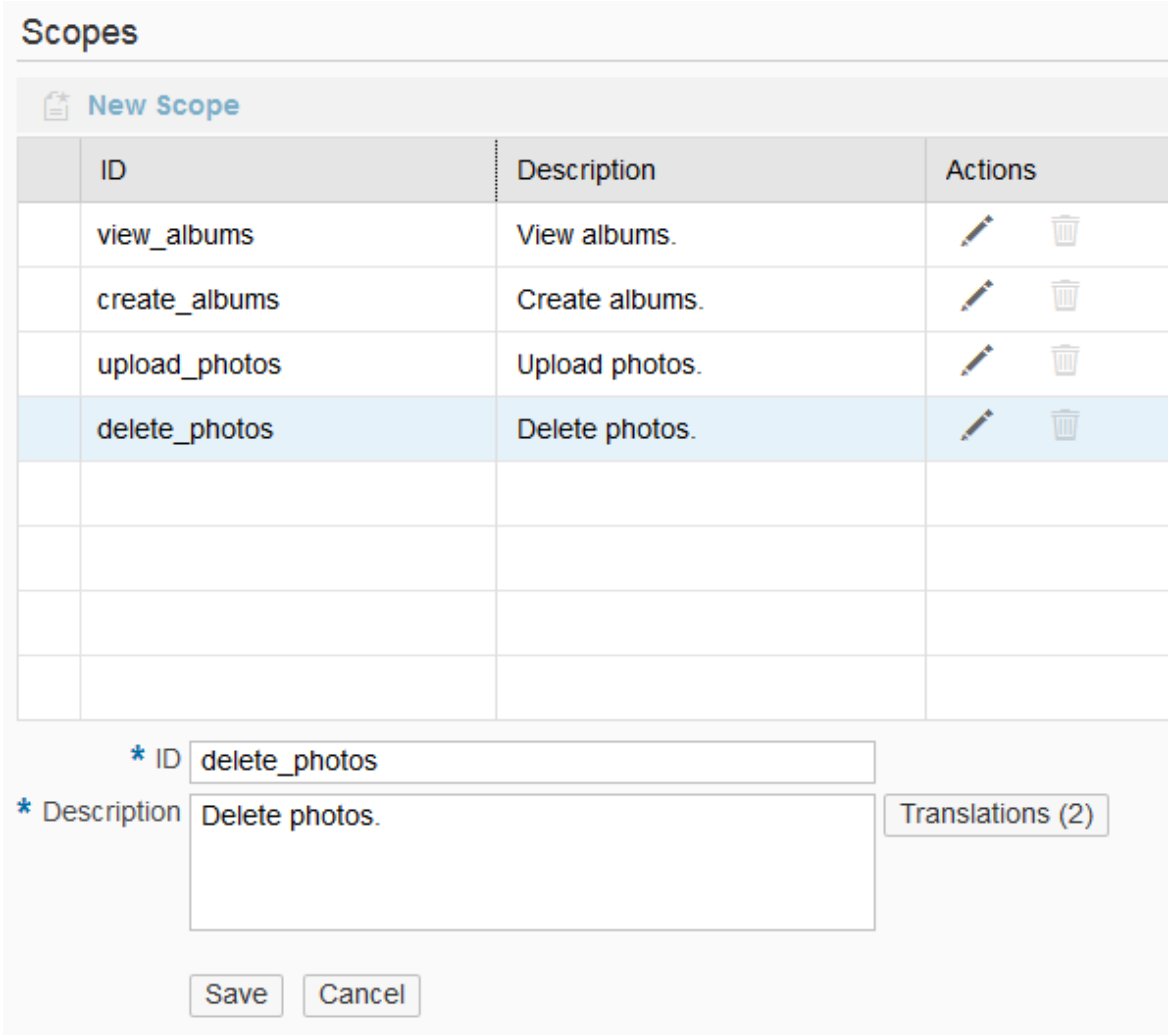
7.3.3.2 Define OAuth Scopes

Define scopes for your OAuth-protected application to fine-grain the access rights to it.

Context

Procedure

1. In your Web browser, log on to the cockpit, and select an account.
2. In the *Applications* > *Java Applications* section, select the OAuth-protected application.
3. For the application, go to the *Security* > *OAuth Scopes* section.
4. Choose *New Scope*.



5. Enter the scope ID and description.
6. Optionally, if you want to provide localization for different languages, choose *Translations* and enter the required data.
7. Save the new scope.

7.3.3.3 Revoke OAuth Access Tokens

With revoking access tokens, you can immediately reject access rights you have previously granted. You may wish to revoke an access token if you believe the token is be stolen, for example.

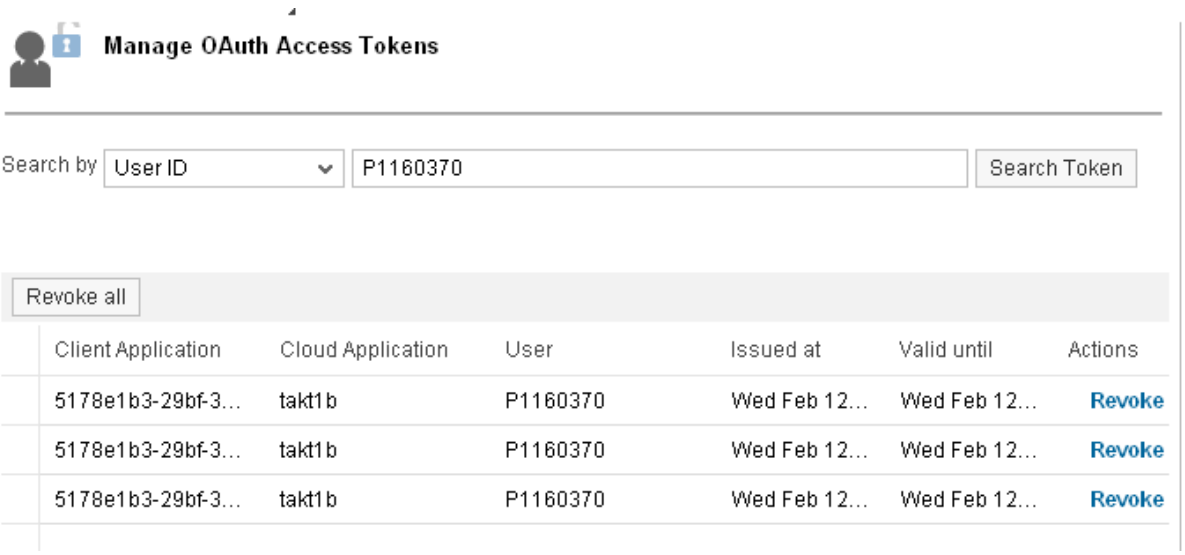
> **! Restriction**
>
> You can revoke only the default persistent type of accesss tokes. If you use JWT tokens as access tokens, you cannot revoke them as they are not stored persistently. See Best Practices for Resilient OAuth 2.0 Communication [page 1768].

There are two UIs for revoking access tokens:

- The Cockpit - an administrator user may use the Cockpit to revoke tokens on behalf of different end users
- The end user UI - an end user may access its tokens (and no other user's) and revoke the required using that UI

Using the Cockpit (for administrators):

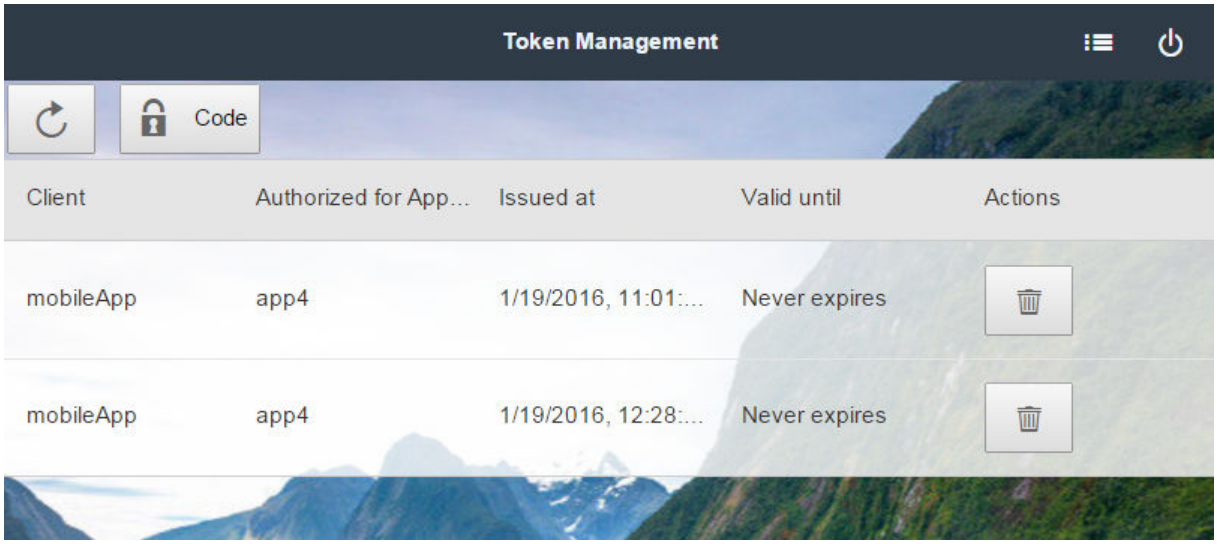
1. In your Web browser, open the Cockpit.
2. Go to the *Security* > *Authorizations* > *Token* section.
3. Search for access tokens either by client ID or by user ID.
4. Choose *Revoke* for the required tokens.



Using the End User UI:

1. In the Cockpit, choose the *Security* > *OAuth* section, and go to the *Branding* tab.

2. Click the *End User UI* link.You are now opening the end user UI in a new browser window. You can see all access tokens issued for the current user.
3. Choose the *Revoke* button for the tokens to revoke.



7.3.3.4 Use a QR Code for Mobile Access

Use a QR code for easier copying of the OAuth authorization code on mobile devices.

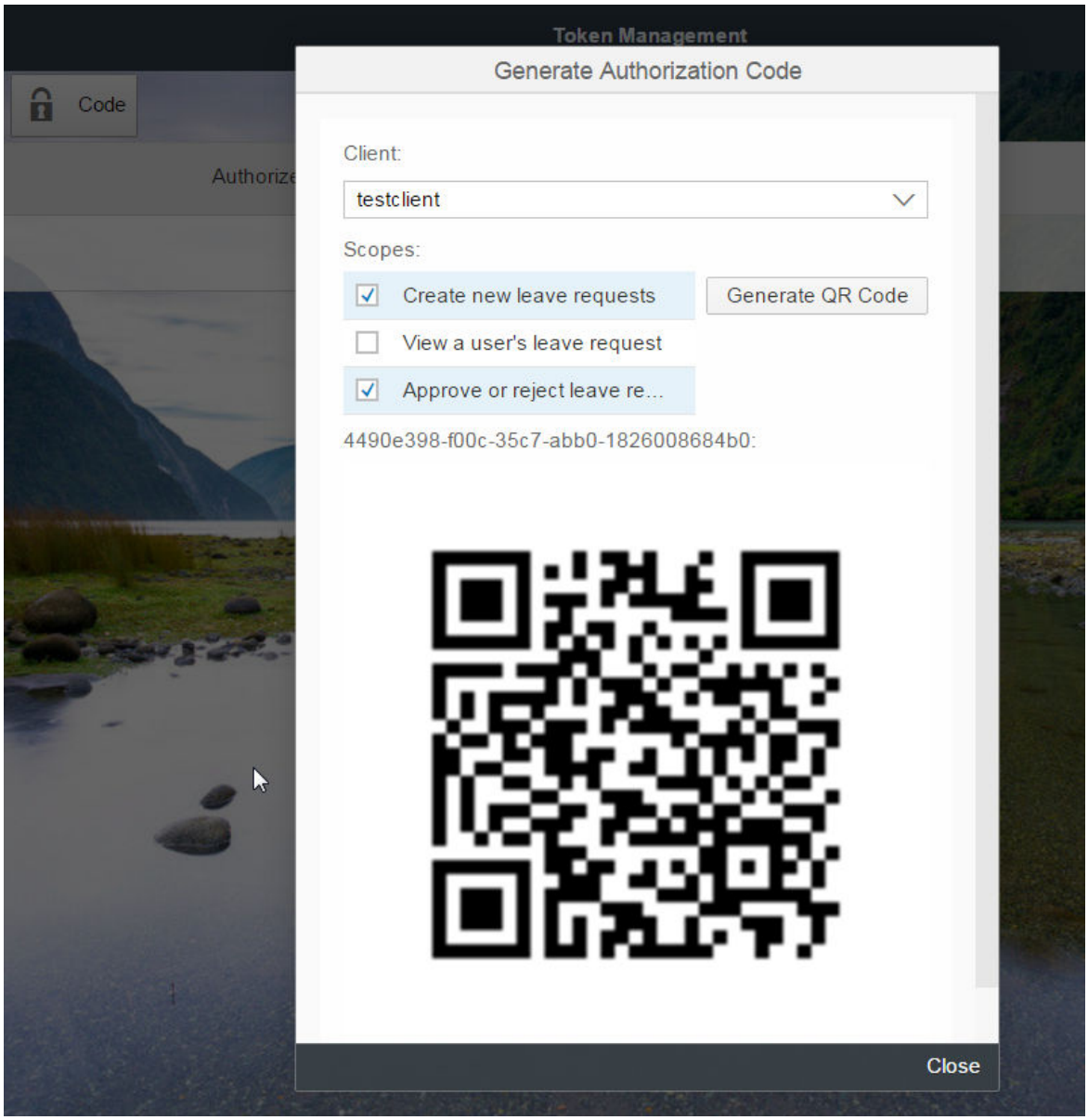
Context

When your account is configured for trust with a corporate identity provider (IdP), it is often impossible to connect to the IdP directly using a personal mobile device. The corporate IdP is often part of a protected corporate network, which does not allow personal devices to access it. To facilitate OAuth authentication on mobile devices, you can use the end user UI's QR code generation option. It provides as a scannable QR code the authorization code sent by the OAuth authorization server.

Procedure

1. In the Cockpit, choose the *Security* > *OAuth* > *Branding* section.
2. Click the *End User UI* link. You are now opening the end user UI in a new browser window.
3. Choose *Code*.
4. Select the client from the list of registered clients for this user.
5. Select the required scopes.
6. Choose *Generate QR Code*.

7. Use your mobile device to scan this QR code (prerequisite: you have QR code scanning software installed), and copy it to your device's clipboard.
8. Paste the code from the clipboard to your mobile application (prerequisite: your mobile application allows you to paste the authorization code from the clipboard and will send in this case the access token request directly to the OAuth authorization server).



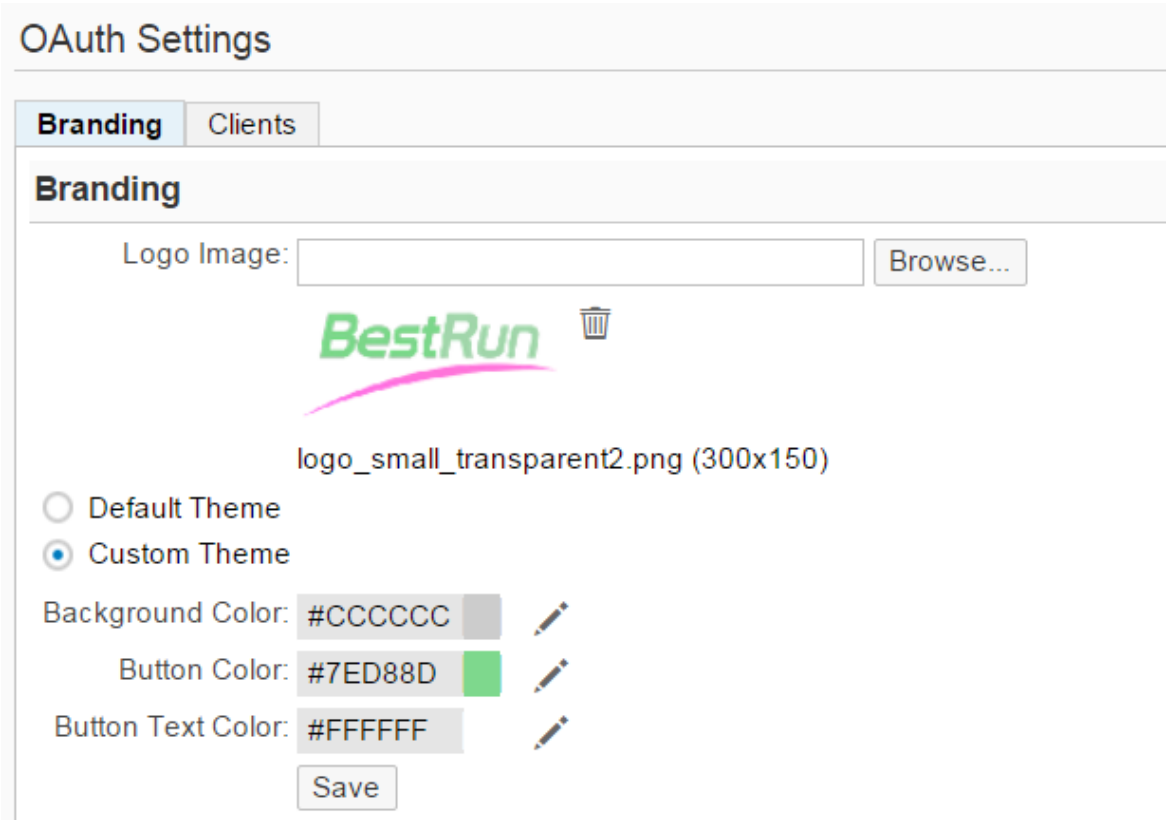
7.3.3.5 Customize Corporate Branding

You can customize the lookandfeel of the authorization page displayed to end users with your corporate branding. This will make it easier for them to recognize your organization.

Context

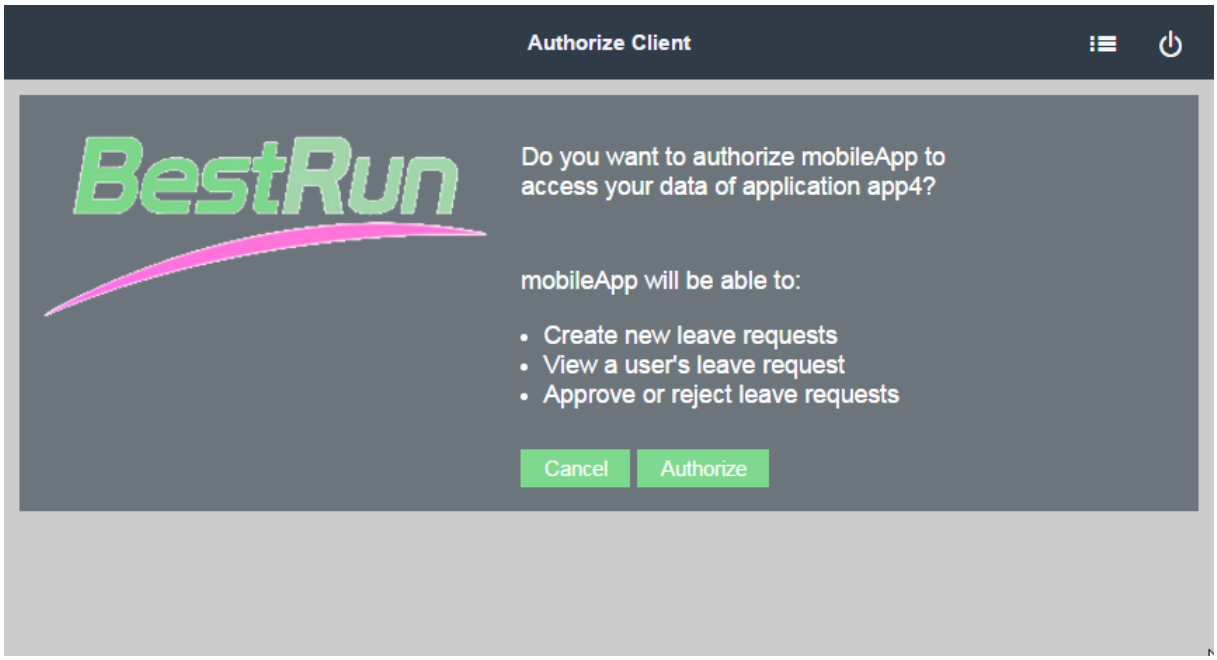
Procedure

1. In your Web browser, log on to the cockpit, and select an account.
2. Go to the *Security* > *OAuth* > *Branding* section.
3. Choose *Custom theme* and specify the required colors.
4. Upload your custom icon.
5. Save the settings.



Results

The authorization page that end users see contains the company logo and colors you specify. The following image shows an example of a customized authorization page.



7.3.4 Best Practices for Resilient OAuth 2.0 Communication

Use the following best practices to build resilient OAuth 2.0 communication.

Key Points

- Tokens can always be issued.
- Persistent tokens cannot be validated in rare cases (for example, DB outage and LRU cache miss). In this case 401 would be returned.
- JWT tokens can always be validated.

JWT Tokens

An OAuth 2.0 Client can request one of the following two types of access tokens: persistent (stored in the system database (default situation)) or JWT (on-demand).

Using JWT tokens enables faster and more robust token validation than using the default persistent tokens. Hence, we recommend that JWT tokens are used whenever possible (see some constraints below).

They can be issued for the as access token for both Client Credentials Flow and Authorization Code Flow. Refresh tokens are always persistent.

Requesting JWT Tokens

Append the following query parameter to the token request:

```
"token_format=jwt"
```

Note: For some Platform APIs, JWT tokens are used by default and cannot be switched off.

JWT Token Constraints

- Their validity is maximum 12 hours or less. If you need a longer token validity, use the default persistent tokens.
- They cannot be listed in the OAuth End User UI or the Cockpit, since they aren't persisted/stored by the OAuth 2.0 Service.
- They cannot be revoked.
- Regardless of the expiration configuration of the OAuth client, if a JWT token is requested, the validity will be 12h or less.
- Their size is in the 2-3kb range, compared to <40 bytes for the default persistent tokens

Best Practices for Resilient OAuth Communication

- Use JWT tokens whenever possible. Use persistent tokens only in the very limited cases when JWT tokens are not applicable.
- Do not rely on the expiration time configured in the OAuth client. It is rather a hint to the OAuth server. The OAuth server may expire tokens earlier in critical situations or validation may be temporarily unavailable. Token issuing is always possible. So in case of a 401, implement a retry with issuing a new access token.
- Ensure that the token used is valid at least a certain amount of time, for example, 1 hour. This will ensure that even if there is some temporary outage, the OAuth client will have a valid token during the outage.
    - This can be achieved by requesting a new token at least 1 hour before the old one is expired. In case the token is successfully retrieved it can be used directly. And in case the OAuth 2.0 Service is unavailable, the old one can still be used.
    - JWT tokens can be validated even if the OAuth 2.0 Service is down or not responding
    - JWT and persistent tokens can always be issued.
- Retry calls for issuing tokens
    - HTTP protocol is unreliable and a call can fail for a multitude of reasons before it does HTTP Response is received
    - Some short delay between calls is always reasonable.
- Reuse tokens instead of issuing a new one every time.

7.3.5 Principal Propagation to OAuth-Protected Applications

Propagate users from external applications with SAML identity federation to OAuth-protected applications running in the Neo environment of SAP BTP. Exchange the user ID and attributes from a SAML assertion for an OAuth access token, and use the access token to access the OAuth-protected application.

Prerequisites

- You have an application external to SAP BTP. The application is integrated with a third-party library or system functioning as a SAML identity provider. That application has a SAML assertion for each authenticated user.

  > **i** Note
  >
  > How the external application and its SAML identity provider work together and communicate is outside the scope of this documentation. They can be separate applications, or the external application may be using a library integrated in it.

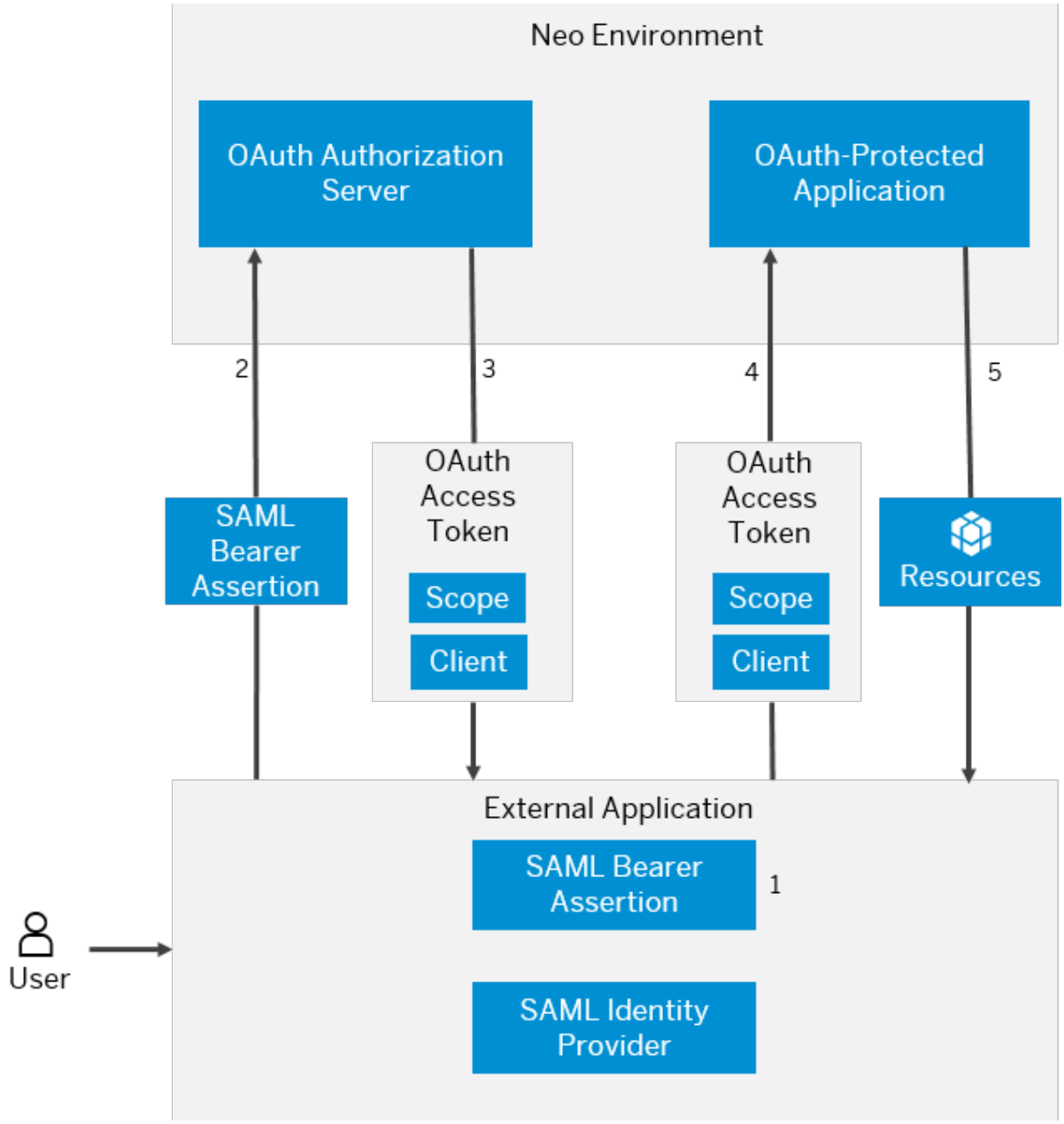
  > **i** Note
  >
  > If you are using a separate third-party identity provider system for this scenario, make sure you have configured correctly trust between the external application and the identity provider system. Refer to the identity provider vendor's documentation for details.

- You have configured SAP BTP for identity federation. See Configure the Local Service Provider [page 1721].
- You have developed an OAuth-protected application. See OAuth 2.0 Authorization Code Grant [page 1753].
- You have deployed the OAuth-protected application at SAP BTP. See Deploying and Updating Applications [page 875].

Context

This scenario follows the SAML 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants specification. The scenario is based on exchanging the **SAML (bearer) assertion** from the third-party identity provider for an **OAuth access token** from the SAP BTP authorization server. Using the access token, the external application can access the OAuth-protected application.

The graphic below illustrates the scenario implemented in terms of SAP BTP.



1. An external application has a SAML assertion on behalf of a successfully logged user. The application needs to proparate that user and its relevant information (attributes, privileges, and so on) to the OAuth-protected application running at SAP BTP.
2. The external application passes the SAML assertion to SAP BTP.
   To access the OAuth-protected application at SAP BTP, however, the external application needs an OAuth 2.0 access token, not a SAML assertion.
3. If the SAML assertion contains all required information (see the procedure below) SAP BTP generates the corresponding access token. In this way, the external application can act on behalf of the user authenticated by the identity provider, within its granted privileges at SAP BTP, and within the time limits of the access token.
4. The external application passes the received access token to the OAuth-protected application at SAP BTP.

5. If the access token is correct and the user has the required privileges, the OAuth-protected application returns the requested resources.

Procedure

1. Configure SAP BTP for trust with the SAML identity provider. See Configure Trust to the SAML Identity Provider [page 1724].
2. Register the external application as an OAuth client in SAP BTP. See Register an OAuth Client [page 1760].
3. Make sure the SAML (bearer) assertion that the external application presents contains the following information:

SAML Assertion Element	Value Description	Example
Name ID	The authenticated user ID.	`<saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid format:unspecified" xmlns:saml="urn:oasis:names:tc:SAML: 2.0:assertion">p12356789 </saml:NameID>`

<table>
<tr><th>SAML Assertion Element</th><th>Value Description</th><th>Example</th></tr>
<tr><td>Audience</td><td>The local service provider name for your SAP BTP account (in Cockpit > Security > Trust > Local Service Provider > Local Provider Name).

If you are using the default identity provider configuration for your account, take the audience value from the table below.

<table>
<tr><th>Landscape Host</th><th>Description</th><th>Required Audience Value</th></tr>
<tr><td>hana.ondemand.com</td><td>Productive landscape, data center Europe</td><td>https://netweaver.ondemand.com</td></tr>
<tr><td>ap1.hana.ondemand.com</td><td>Productive landscape, data center Asia-Pasific (Australia)</td><td>ap1.hana.ondemand.com</td></tr>
<tr><td>https://us1.hana.ondemand.com</td><td>Productive landscape, data center United States (US East)</td><td>https://us1.hana.ondemand.com/</td></tr>
<tr><td>hana-trial.ondemand.com</td><td>Trial landscape</td><td>https://nwtrial.ondemand.com</td></tr>
</table>

See Regions.</td><td><pre><saml:Audience>myLocalProvider</saml:Audience></pre>
<pre><saml:Audience>https://us1.hana.ondemand.com/</saml:Audience></pre>
<pre><saml:Audience>ap1.hana.ondemand.com</saml:Audience></pre></td></tr>
</table>

SAML Assertion Element	Value Description	Example

Issuer ID	The issuer must have as value the OAuth client ID registered at SAP BTP (in *Cockpit* > *Security* > *OAuth* > *Clients* > *<your client>* > *Client ID*).	`<saml:Issuer`
		
		`Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity"`
		
		`xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">myClientID`
		`</saml:Issuer>`

Issuer Certificate	The identity provider signing certificate stored in the trust configuration of SAP BTP for this identity provider (in *Cockpit* > *Security* > *Trust* > *Application Identity Provider* > *<your identity provider* > *General* > *Signing Certificate*).	Sample Code
		`<ds:X509Certificate>`
		`…… </ds:X509Certificate>`

(Optional) User Attributes	The attributes that will be assigned to the SAP BTP user.	`<Attribute Name="mail">`
		`<AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"`
		
		`xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`
		
		`xsi:type="xs:string">test@sap.com`
		`</AttributeValue>`
		`</Attribute>`
		`<Attribute Name="first_name">`
		`<AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"`
		
		`xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`
		
		`xsi:type="xs:string">Jon`
		`</AttributeValue>`
		`</Attribute>`

See the SAML Assertion Format specification for more information.

4. In the code of the OAuth-protected application, you can retrieve the user attributes using the relevant SAP BTP API. See User Attributes [page 1696].

7.3.6 Auditing and Logging Information for OAuth 2.0 Service

Here you can find a list of the security events that are logged by OAuth 2.0 Service..

Security events written in audit logs

Event grouping	What events are logged	How to identify related log events	Additional information
Authorization code	Issue OAuth authorization code	"action":"create" & "key":"Creating authorization code: <first 5 symbols of the code>" Distincted by tenantId:"<tenant id>"	
	Delete OAuth authorization code	"action":"delete" & "key":"Deleting authorization code: <first 5 symbols of the code>" Distincted by tenantId:"<tenant id>"	
Access token	Issue access token for client credentials flow	"action":"create" & "key":" Creating access token: <first 5 symbols of the token>" "action":"create" & "key":"Creating platform access token: <first 5 symbols of the token>" - when using platform client Distincted by tenantId:"<tenant id>"	
	Issue access token from authorization code	"action":"create" & "key":"Creating access token: <first 5 symbols of the token>" Distincted by tenantId:"<tenant id>"	
	Issue access token from refresh token	"action":"create" & "key":"Creating access token: <first 5 symbols of the token>" & "key":"Creating refresh token: <first 5 symbols of the token>" Distincted by tenantId:"<tenant id>"	

Event grouping	What events are logged	How to identify related log events	Additional information
	Issue acccess token from SAML bearer assertion	"action":"create" & "objectID":"Creating access token: <first 5 symbols of the token>" Distincted by tenantId:"<tenant id>" Further refining of search: "operation":"Create access token from SAML bearer"	
	Delete access token	"action":"delete" & "key":"Deleting cloud access token: <first 5 symbols of the token>" Distincted by tenantId:"<tenant id>"	
Refresh token	Create refresh token	"action":"create" & "key":""Creating refresh token: " <first 5 symbols of the token>" Distincted by tenantId:"<tenant id>"	
OAuth client	Create OAuth client	"action":"create" & "key":"Creating application client: <client id>" Distincted by tenantId:"<tenant id>"	
	Update OAuth client	"action":"update" & "key":"Updating application client: <client id>" Distincted by tenantId:"<tenant id>"	
	Delete OAuth client	"action":"delete" & "key":"Deleting application client: <client id>" Distincted by tenantId:"<tenant id>"	
Platform client	Create OAuth platform client	"action":"create" & "key":"Creating platform client: <client id>" Distincted by "account":"<account>"	

Event grouping	What events are logged	How to identify related log events	Additional information
	Delete OAuth platform client	"action":"delete" & "key":"Deleting platform client: <client id>" Distincted by "account":"<account>"	
	Create multi-tenant OAuth platform client	"action":"create" & "objectID":"Creating platform client: <client id>" Distincted by "account":"<account>" Further refining of search: "operation":"Create Multitenant Platform API client"	
	Create admin OAuth platform client	"action":"create" & "key":"Creating platform client: <client id>" Distincted by: • "account":"<ips_global_account>" • "account":"<ias_resource_server>	
	Delete admin OAuth platform client	"action":"delete" & "key":"Deleting platform client: <client id>" Distincted by: • "account":"<ips_global_account>" • "account":"<ias_resource_server>	

Event grouping	What events are logged	How to identify related log events	Additional information
	Create external OAuth platform client	"action":"create" & "objectID":"Creating platform client: <client id>" Distincted by: • "account":"<ips_global_account>" • "account":"<ias_resource_server> Further refining of search: "operation":"Delete External Platform API client"	
	Delete external OAutth platform client	"action":"delete" & "objectID":"Deleting platform client: <client id>" Distincted by: • "account":"<ips_global_account>" • "account":"<ias_resource_server> Further refining of search: "operation":"Delete External Platform API client"	

The following information is described in the table columns:

- *Event grouping* - Events that are logged with a similar format or are related to the same entities.
- *What events are logged* - Description of the security or data protection and privacy related event that is logged.
- *How to identify related log events* - Search criteria or key words, that are specific for a log event that is created along with the logged event.
- *Additional information* - Any related information that can be helpful.

Related Information

Audit Logging in the Cloud Foundry Environment
Audit Logging in the Neo Environment

7.4 Keystore Service

The Keystore Service provides a repository for cryptographic keys and certificates to the applications in the Neo environment of SAP BTP.

If you want to use cryptography with unlimited strength in an SAP BTP application, you need to enable it via installing the necessary Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files on SAP JVM.

Parent topic: Security, Neo Environment [page 1671]

Related Information

Authorization and Trust Management in the Neo Environment [page 1672]
Platform Identity Provider [page 1746]
OAuth 2.0 Service [page 1751]
Audit Logging in the Neo Environment [page 1810]
Principal Propagation [page 1816]
Protection from Web Attacks [page 1835]
Data Protection and Privacy [page 1848]
Keys and Certificates [page 1781]
Enabling Client Certificate Authentication [page 1797]
Enable Strong Encryption in Applications [page 1807]
Storing Passwords [page 1808]

7.4.1 Keystore API

The Keystore API provides a repository for cryptographic keys and certificates to the applications in the Neo environment. It allows you to manage keystores at subaccount, application or subscription level.

The Keystore API is protected with *OAuth 2.0 client credentials*. Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149].

Using an HTTP Destination for Connecting to the Keystore

Using an HTTP destination is a convenient way to establish connection to the keystore. Once created, you can re-use the destination for different API calls. To create the required destination, do the following steps:

1. At the required level, create an HTTP destination with the following information:
   - Name=<your destination name>
   - URL=https://api.<cloud platform host>/keystore/v1
   - ProxyType=Internet
   - Type=HTTP
   - CloudConnectorVersion=2
   - Authentication=NoAuthentication

   See Create HTTP Destinations [page 80].
2. In your application, obtain an `HttpURLConnection` object that uses the destination.
   See ConnectivityConfiguration API [page 139].

Example: Downloading a Keystore at Account Level

With the `HttpURLConnection` object retrieved, execute a GET call.

```
//value is the URL property of the destination
URL url = new URL(value + "accounts/{account}/{keystoreName}");
HttpURLConnection urlConnection = (HttpURLConnection)url.openConnection();
urlConnection.setRequestMethod("GET");
urlConnection.setRequestProperty("Authorization", "Bearer <OAuth token ID from
previous step>");
urlConnection.connect();
```

If everything is OK, you should get a response like that:

```
200 - <content of the keystore>
```

> → Tip
>
> We recommend using `If-None-Match` header for subsequent calls to the keystore to check if the keystore contents have been modified since your last GET call.
>
> From the response, copy the `Etag` header value and repeat the request with the added header below:
>
> ```
> If-None-Match : <Etag_header value from the first response>
> ```
>
> You can do it using the same code excerpt as above with added the following line before the last line:
>
> ```
> urlConnection.setRequestProperty("If-None-Match", "<value of Etag header>");
> ```
>
> Expected responses:
>
> - If no content has been modified:
>   `304 - Not Modified`
> - If the content was successfully modified:
>   `200 - <the modified keystore content>`

Example: Uploading a Keystore at Account Level

With the `HttpURLConnection` object retrieved, execute a PUT call.

```
URL url = new URL(value + "accounts/{account}/{keystoreName}");
HttpURLConnection urlConnection = (HttpURLConnection)url.openConnection();
urlConnection.setRequestMethod("PUT");
urlConnection.setRequestProperty("Authorization",  "Bearer < token ID from
previous step>");
urlConnection.setRequestProperty("Content-Type",  "<selected_content_type>");
urlConnection.setDoOutput(true);
urlConnection. getOutputStream().write(keyStore.getBytes());  // writes the
keystore bytes in the URL connection
urlConnection.connect();
```

If you want to overwrite the keystore, set to true the query parameter `overwrite`. For example:

```
URL url = new URL(value + "accounts/{account}/{keystoreName}?overwrite=true");
```

The `Content-Type` header can be:

- `application/x-pkcs12` - for PKCS12 formatted keystores
  `application/x-pem-file` - for PEM formatted keystores
  `application/octet-stream` – for any other type.

Expected response:

- If the keystore upload was successful:
  `200 - OK`
- If the keystore already exists:
  `409 - conflict`

Related Information

Using Platform APIs [page 1149]
Keystore API

7.4.2 Keys and Certificates

Overview

The Keystore Service provides a repository for cryptographic keys and certificates to the applications hosted on SAP BTP. By using the Keystore Service, the applications could easily retrieve keystores and use them in various cryptographic operations such as signing and verifying of digital signatures, encrypting and decrypting messages, and performing SSL communication.

Features

The SAP HANA Keystore Service stores and provides keystores encoded in the following formats:

- Java Keystore (JKS)
  It supports password-based integrity validation for its content. Key entries are protected with password-based encryption. Password has to be specified in order to retrieve a key entry.
- Extended Java Keystore (JCEKS)
  It provides the same functionality as the JKS format with stronger protection for private keys.
- PKCS #12 file (P12)
  This format supports password-based integrity validation for its content. Key entries are protected with password-based symmetric encryption. A password has to be specified in order to retrieve a key entry.
- Privacy Enhanced Mail Certificate (PEM)
  It does not support integrity validation. Key entries are not protected with password.

Configuring Keystores

- Local Server Configuration
  You can manage the keystores directly on the file system of the local server. Place the keystore files with **.jks**, **.pem**, **.jceks**, or **.p12** extension in the following folder: `<local server>/config_master/com.sap.cloud.crypto.keystore`.
- Cloud Configuration
  You can manage the keystores via the SAP BTP console client. For more information, see Keystore Console Commands [page 1783].

Keystore Search Order

The keystore service works with keystores available on the following levels:

- Subscription level
  Keystores available for a certain application provided by another account.
- Application level
  Keystores available for a certain application in a particular consumer account.
- Account level
  Keystores available for all applications in a particular consumer account.

When searching for a keystore with a certain name, the keystore service will search on the different levels in following order: *Subscription level* > *Application level* > *Account level*.

Once a keystore with the specified name has been found at a certain location, further locations will no more be searched for.

Consuming the Keystore Service

To consume the Keystore Service, you need to add the following reference to your `web.xml` file:

```
<resource-ref>
   <res-ref-name>KeyStoreService</res-ref-name>
   <res-type>com.sap.cloud.crypto.keystore.api.KeyStoreService</res-type>
</resource-ref>
```

Then, in the code you can look up Keystore Service API via JNDI:

```
import com.sap.cloud.crypto.keystore.api.KeyStoreService;
...
KeyStoreService keystoreService = (KeyStoreService) new
InitialContext().lookup("java:comp/env/KeyStoreService");
```

A keystore can be obtained by using the `getKeyStore()` method.

```
KeyStore keyStore = keystoreService.getKeyStore(keystoreName, keystorePassword);
```

For more information, see *Tutorial: Using the Keystore Service for Client Side HTTPS Connections*.

Related Information

Keystore Console Commands [page 1783]
Using the Keystore Service for Client Side HTTPS Connections [page 1791]

7.4.2.1 Keystore Console Commands

The keystore console commands are called from the SAP BTP console client and allow users to list, upload, download, and delete keystores. To be able to use them, the user must have administrative rights for that account. The console supports the following keystore commands: **list-keystores**, **upload-keystore**, **download-keystore**, and **delete-keystore**.

Related Information

Keys and Certificates [page 1781]
list-keystores [page 1493]
upload-keystore [page 1581]
download-keystore [page 1439]
delete-keystore [page 1413]

7.4.2.2 Trusted Certificate Authorities for Outbound SSL Connections

List of certificate authorities trusted by the virtual machines running the applications.

The virtual machines for applications trust the below-listed certificate authorities (CAs) by default. This means that the external HTTPS services which use X.509 server certificates (which are issued by those CAs), are trusted by default. No trust needs to be configured manually.

For SSL connections to services which use different certificate issuers, you need to configure trust to use the keystore service of the platform. For more information, see Using the Keystore Service for Client Side HTTPS Connections [page 1791].

Trusted certificate authorities:

Certificate Alias	Certificate Name	Certificate SHA1
actalisauthenticationrootca [jdk]	CN=Actalis Authentication Root CA, O=Actalis S.p.A./03358520967, L=Milan, C=IT	F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
affirmtrustcommercialca [jdk]	CN=AffirmTrust Commercial, O=AffirmTrust, C=US	F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
affirmtrustnetworkingca [jdk]	CN=AffirmTrust Networking, O=AffirmTrust, C=US	29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
affirmtrustpremiumca [jdk]	CN=AffirmTrust Premium, O=AffirmTrust, C=US	D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
affirmtrustpremiumeccca [jdk]	CN=AffirmTrust Premium ECC, O=AffirmTrust, C=US	B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
amazonrootca1 [jdk]	CN=Amazon Root CA 1, O=Amazon, C=US	8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
amazonrootca2 [jdk]	CN=Amazon Root CA 2, O=Amazon, C=US	5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
amazonrootca3 [jdk]	CN=Amazon Root CA 3, O=Amazon, C=US	0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
amazonrootca4 [jdk]	CN=Amazon Root CA 4, O=Amazon, C=US	F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
atostrustedroot2011	C=DE, O=Atos, CN=Atos TrustedRoot 2011	2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
baltimorecybertrustca	CN=Baltimore CyberTrust Root, OU=CyberTrust, O=Baltimore, C=IE	D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
buypassclass2ca [jdk]	CN=Buypass Class 2 Root CA, O=Buypass AS-983163327, C=NO	49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
buypassclass3ca [jdk]	CN=Buypass Class 3 Root CA, O=Buypass AS-983163327, C=NO	DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57

Certificate Alias	Certificate Name	Certificate SHA1
camerfirmachambersca [jdk]	CN=Chambers of Commerce Root - 2008, O=AC Camerfirma S.A., SERIALNUMBER=A82743287, L=Madrid (see current address at www.camerfirma.com/address), C=EU	78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
camerfirmachamberscommerceca [jdk]	CN=Chambers of Commerce Root, OU=http://www.chambersign.org, O=AC Camerfirma SA CIF A82743287, C=EU	6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
camerfirmachambersignca [jdk]	CN=Global Chambersign Root - 2008, O=AC Camerfirma S.A., SERIALNUMBER=A82743287, L=Madrid (see current address at www.camerfirma.com/address), C=EU	4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
certumca [jdk]	CN=Certum CA, O=Unizeto Sp. z o.o., C=PL	62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
certumtrustednetworkca [jdk]	CN=Certum Trusted Network CA, OU=Certum Certification Authority, O=Unizeto Technologies S.A., C=PL	07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
chunghwaepkirootca [jdk]	OU=ePKI Root Certification Authority, O="Chunghwa Telecom Co., Ltd.", C=TW	67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
comodoaaaca [jdk]	CN=AAA Certificate Services, O=Comodo CA Limited, L=Salford, ST=Greater Manchester, C=GB	D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
comodoeccca [jdk]	CN=COMODO ECC Certification Authority, O=COMODO CA Limited, L=Salford, ST=Greater Manchester, C=GB	9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
comodorsaca	CN=COMODO RSA Certification Authority, O=COMODO CA Limited, L=Salford, ST=Greater Manchester, C=GB	AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
digicertassuredidg2 [jdk]	CN=DigiCert Assured ID Root G2, OU=www.digicert.com, O=DigiCert Inc, C=US	A1:4B:48:D9:43:EE:0A:0E:40:90:4F:3C:E0:A4:C0:91:93:51:5D:3F
digicertassuredidg3 [jdk]	CN=DigiCert Assured ID Root G3, OU=www.digicert.com, O=DigiCert Inc, C=US	F5:17:A2:4F:9A:48:C6:C9:F8:A2:00:26:9F:DC:0F:48:2C:AB:30:89
digicertassuredidrootca	CN=DigiCert Assured ID Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US	05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
digicertglobalrootca_g2	CN=DigiCert Global Root G2, OU=www.digicert.com, O=DigiCert Inc, C=US	DF:3C:24:F9:BF:D6:66:76:1B:26:80:73:FE:06:D1:CC:8D:4F:82:A4
digicertglobalrootcalss2_g3	CN=DigiCert Global Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US	A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36

Certificate Alias	Certificate Name	Certificate SHA1
digicertglobalrootg3 [jdk]	CN=DigiCert Global Root G3, OU=www.digicert.com, O=DigiCert Inc, C=US	7E:04:DE:89:6A:3E:66:6D:00:E6:87:D3:3F:FA:D9:3B:E8:3D:34:9E
digicerthighassuranceevrootca	CN=DigiCert High Assurance EV Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US	5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
digicerttrustedrootg4 [jdk]	CN=DigiCert Trusted Root G4, OU=www.digicert.com, O=DigiCert Inc, C=US	DD:FB:16:CD:49:31:C9:73:A2:03:7D:3F:C8:3A:4D:7D:77:5D:05:E4
dtrustclass3ca2 [jdk]	CN=D-TRUST Root Class 3 CA 2 2009, O=D-Trust GmbH, C=DE	58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
dtrustclass3ca2ev [jdk]	CN=D-TRUST Root Class 3 CA 2 EV 2009, O=D-Trust GmbH, C=DE	96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
entrust_ca_2048	CN=Entrust.net Certification Authority (2048), OU=(c) 1999 Entrust.net Limited, OU=www.entrust.net/CPS_2048 incorp. by ref. (limits liab.), O=Entrust.net	50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
entrust_ev_ca	CN=Entrust Root Certification Authority, OU="(c) 2006 Entrust, Inc.", OU=www.entrust.net/CPS is incorporated by reference, O="Entrust, Inc.", C=US	B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
entrust_g2_ca	CN=Entrust Root Certification Authority - G2, OU="(c) 2009 Entrust, Inc. - for authorized use only", OU=See www.entrust.net/legal-terms, O="Entrust, Inc.", C=US	8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
entrustrootcaec1 [jdk]	CN=Entrust Root Certification Authority - EC1, OU="(c) 2012 Entrust, Inc. - for authorized use only", OU=See www.entrust.net/legal-terms, O="Entrust, Inc.", C=US	20:D8:06:40:DF:9B:25:F5:12:25:3A:11:EA:F7:59:8A:EB:14:B5:47
entrustrootcag4 [jdk]	CN=Entrust Root Certification Authority - G4, OU="(c) 2015 Entrust, Inc. - for authorized use only", OU=See www.entrust.net/legal-terms, O="Entrust, Inc.", C=US	14:88:4E:86:26:37:B0:26:AF:59:62:5C:40:77:EC:35:29:BA:96:01
gd-class2-root	OU=Go Daddy Class 2 Certification Authority, O="The Go Daddy Group, Inc.", C=US	27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
gd_intermediate	SERIALNUMBER=07969287, CN=Go Daddy Secure Certification Authority, OU=http://certificates.godaddy.com/repository, O="GoDaddy.com, Inc.", L=Scottsdale, ST=Arizona, C=US	7C:46:56:C3:06:1F:7F:4C:0D:67:B3:19:A8:55:F6:0E:BC:11:FC:44

Certificate Alias	Certificate Name	Certificate SHA1
gdroot-g2	CN=Go Daddy Root Certificate Authority - G2, O="GoDaddy.com, Inc.", L=Scottsdale, ST=Arizona, C=US	47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
geotrust_pca_g3_root	CN=GeoTrust Primary Certification Authority - G3, OU=(c) 2008 GeoTrust Inc. - For authorized use only, O=GeoTrust Inc., C=US	03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
geotrustglobalca	CN=GeoTrust Global CA, O=GeoTrust Inc., C=US	DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
geotrustprimaryca	CN=GeoTrust Primary Certification Authority, O=GeoTrust Inc., C=US	32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
geotrustprimarycag2 [jdk]	CN=GeoTrust Primary Certification Authority - G2, OU=(c) 2007 GeoTrust Inc. - For authorized use only, O=GeoTrust Inc., C=US	8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
geotrustuniversalca	CN=GeoTrust Universal CA, O=GeoTrust Inc., C=US	E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
globalsignca	CN=GlobalSign Root CA, OU=Root CA, O=GlobalSign nv-sa, C=BE	B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
globalsigneccrootcar4 [jdk]	CN=GlobalSign, O=GlobalSign, OU=GlobalSign ECC Root CA - R4	69:69:56:2E:40:80:F4:24:A1:E7:19:9F:14:BA:F3:EE:58:AB:6A:BB
globalsigneccrootcar5 [jdk]	CN=GlobalSign, O=GlobalSign, OU=GlobalSign ECC Root CA - R5	1F:24:C6:30:CD:A4:18:EF:20:69:FF:AD:4F:DD:5F:46:3A:1B:69:AA
globalsigneccrootcar6	CN=GlobalSign, O=GlobalSign, OU=GlobalSign Root CA - R6	80:94:64:0E:B5:A7:A1:CA:11:9C:1F:DD:D5:9F:81:02:63:A7:FB:D1
globalsignr2ca	CN=GlobalSign, O=GlobalSign, OU=GlobalSign Root CA - R2	75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
globalsignr3ca	CN=GlobalSign, O=GlobalSign, OU=GlobalSign Root CA - R3	D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
haricaeccrootca2015 [jdk]	CN=Hellenic Academic and Research Institutions ECC RootCA 2015, O=Hellenic Academic and Research Institutions Cert. Authority, L=Athens, C=GR	9F:F1:71:8D:92:D5:9A:F3:7D:74:97:B4:BC:6F:84:68:0B:BA:B6:66
haricarootca2015 [jdk]	CN=Hellenic Academic and Research Institutions RootCA 2015, O=Hellenic Academic and Research Institutions Cert. Authority, L=Athens, C=GR	01:0C:06:95:A6:98:19:14:FF:BF:5F:C6:B0:B6:95:EA:29:E9:12:A6
identrustcommercial [jdk]	CN=IdenTrust Commercial Root CA 1, O=IdenTrust, C=US	DF:71:7E:AA:4A:D9:4E:C9:55:84:99:60:2D:48:DE:5F:BC:F0:3A:25
identrustdstx3 [jdk]	CN=DST Root CA X3, O=Digital Signature Trust Co.	DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13

Certificate Alias	Certificate Name	Certificate SHA1
identrustpublicca [jdk]	CN=IdenTrust Public Sector Root CA 1, O=IdenTrust, C=US	BA:29:41:60:77:98:3F:F4:F3:EF:F2:31:05:3B:2E:EA:6D:4D:45:FD
letsencryptisrgx1 [jdk]	CN=ISRG Root X1, O=Internet Security Research Group, C=US	CA:BD:2A:79:A1:07:6A:31:F2:1D:25:36:35:CB:03:9D:43:29:A5:E8
luxtrustglobalroot2ca [jdk]	CN=LuxTrust Global Root 2, O=LuxTrust S.A., C=LU	1E:0E:56:19:0A:D1:8B:25:98:B2:04:44:FF:66:8A:04:17:99:5F:3F
quovadisrootca1g3 [jdk]	CN=QuoVadis Root CA 1 G3, O=QuoVadis Limited, C=BM	1B:8E:EA:57:96:29:1A:C9:39:EA:B8:0A:81:1A:73:73:C0:93:79:67
quovadisrootca2 [jdk]	CN=QuoVadis Root CA 2, O=QuoVadis Limited, C=BM	CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
quovadisrootca2g3 [jdk]	CN=QuoVadis Root CA 2 G3, O=QuoVadis Limited, C=BM	09:3C:61:F3:8B:8B:DC:7D:55:DF:75:38:02:05:00:E1:25:F5:C8:36
quovadisrootca3 [jdk]	CN=QuoVadis Root CA 3, O=QuoVadis Limited, C=BM	1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
quovadisrootca3g3 [jdk]	CN=QuoVadis Root CA 3 G3, O=QuoVadis Limited, C=BM	48:12:BD:92:3C:A8:C4:39:06:E7:30:6D:27:96:E6:A4:CF:22:2E:7D
sapcloudrootca	CN=SAP Cloud Root CA, O=SAP SE, L=Walldorf, C=DE	6D:80:92:77:4A:F2:D5:ED:AE:3A:5C:99:D6:56:93:1C:21:97:A9:50
sapglobalrootca [jdk]	CN=SAP Global Root CA, O=SAP AG, L=Walldorf, C=DE	0A:B6:2A:F4:7F:E5:59:84:7D:79:8A:1F:C4:E1:7F:67:FD:7E:82:4C
secomscrootca1 [jdk]	OU=Security Communication RootCA1, O=SECOM Trust.net, C=JP	36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
secomscrootca2 [jdk]	OU=Security Communication RootCA2, O="SECOM Trust Systems CO.,LTD.", C=JP	5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
securetrustca [jdk]	CN=SecureTrust CA, O=SecureTrust Corporation, C=US	87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
sslrooteccca [jdk]	CN=SSL.com Root Certification Authority ECC, O=SSL Corporation, L=Houston, ST=Texas, C=US	C3:19:7C:39:24:E6:54:AF:1B:C4:AB:20:95:7A:E2:C3:0E:13:02:6A
sslrootevrsaca [jdk]	CN=SSL.com EV Root Certification Authority RSA R2, O=SSL Corporation, L=Houston, ST=Texas, C=US	74:3A:F0:52:9B:D0:32:A0:F4:4A:83:CD:D4:BA:A9:7B:7C:2E:C4:9A
sslrootrsaca [jdk]	CN=SSL.com Root Certification Authority RSA, O=SSL Corporation, L=Houston, ST=Texas, C=US	B7:AB:33:08:D1:EA:44:77:BA:14:80:12:5A:6F:BD:A9:36:49:0C:BB
starfieldclass2ca [jdk]	OU=Starfield Class 2 Certification Authority, O="Starfield Technologies, Inc.", C=US	AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
starfieldrootg2ca [jdk]	CN=Starfield Root Certificate Authority - G2, O="Starfield Technologies, Inc.", L=Scottsdale, ST=Arizona, C=US	B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E

Certificate Alias	Certificate Name	Certificate SHA1
starfieldservicesrootg2ca [jdk]	CN=Starfield Services Root Certificate Authority - G2, O="Starfield Technologies, Inc.", L=Scottsdale, ST=Arizona, C=US	92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
swisssigngoldg2ca	CN=SwissSign Gold CA - G2, O=SwissSign AG, C=CH	D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
swisssignplatinumg2ca	CN=SwissSign Platinum CA - G2, O=SwissSign AG, C=CH	56:E0:FA:C0:3B:8F:18:23:55:18:E5:D3:11:CA:E8:C2:43:31:AB:66
swisssignsilverg2ca	CN=SwissSign Silver CA - G2, O=SwissSign AG, C=CH	9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
symantecclass3	CN=Symantec Class 3 Secure Server SHA256 SSL CA, OU=Symantec Trust Network, O=Symantec Corporation, C=US	E7:32:73:E5:3A:CF:E8:0F:41:0B:3E:F4:6B:18:02:87:A0:04:40:CD
teliasonerarootcav1 [jdk]	CN=TeliaSonera Root CA v1, O=TeliaSonera	43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
thawteclass3	OU=Class 3 Public Primary Certification Authority, O="VeriSign, Inc.", C=US	A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
thawteprimaryrootcag2 [jdk]	CN=thawte Primary Root CA - G2, OU="(c) 2007 thawte, Inc. - For authorized use only", O="thawte, Inc.", C=US	AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
thawteprimaryrootcag3	CN=thawte Primary Root CA - G3, OU="(c) 2008 thawte, Inc. - For authorized use only", OU=Certification Services Division, O="thawte, Inc.", C=US	F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
thawteprimaryrootcag4	CN=thawte Primary Root CA - G4, OU="(c) 2012 thawte, Inc. - For authorized use only", OU=Certification Services Division, O="thawte, Inc.", C=US	FA:7C:FB:B2:47:42:77:63:43:1B:7E:6D:75:81:2A:49:CC:8D:30:E4
thawteroot	CN=thawte Primary Root CA, OU="(c) 2006 thawte, Inc. - For authorized use only", OU=Certification Services Division, O="thawte, Inc.", C=US	91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
ttelesecglobalrootclass2ca [jdk]	CN=T-TeleSec GlobalRoot Class 2, OU=T-Systems Trust Center, O=T-Systems Enterprise Services GmbH, C=DE	59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
ttelesecglobalrootclass3ca [jdk]	CN=T-TeleSec GlobalRoot Class 3, OU=T-Systems Trust Center, O=T-Systems Enterprise Services GmbH, C=DE	55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
usertrusteccca [jdk]	CN=USERTrust ECC Certification Authority, O=The USERTRUST Network, L=Jersey City, ST=New Jersey, C=US	D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0

Certificate Alias	Certificate Name	Certificate SHA1
usertrustrsaca [jdk]	CN=USERTrust RSA Certification Authority, O=The USERTRUST Network, L=Jersey City, ST=New Jersey, C=US	2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
verisignclass1_g1	OU=Class 1 Public Primary Certification Authority, O="VeriSign, Inc.", C=US	90:AE:A2:69:85:FF:14:80:4C:43:49:52:EC:E9:60:84:77:AF:55:6F
verisignclass2_g2	OU=VeriSign Trust Network, OU="(c) 1998 VeriSign, Inc. - For authorized use only", OU=Class 2 Public Primary Certification Authority - G2, O="VeriSign, Inc.", C=US	B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
verisignclass3_g1	OU=Class 3 Public Primary Certification Authority, O="VeriSign, Inc.", C=US	74:2C:31:92:E6:07:E4:24:EB:45:49:54:2B:E1:BB:C5:3E:61:74:E2
verisignclass3_g2	OU=VeriSign Trust Network, OU="(c) 1998 VeriSign, Inc. - For authorized use only", OU=Class 3 Public Primary Certification Authority - G2, O="VeriSign, Inc.", C=US	85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
verisignclass3_g3	CN=VeriSign Class 3 Public Primary Certification Authority - G3, OU="(c) 1999 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
verisignclass3_g5	CN=VeriSign Class 3 Public Primary Certification Authority - G5, OU="(c) 2006 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
verisignclass3g4ca [jdk]	CN=VeriSign Class 3 Public Primary Certification Authority - G4, OU="(c) 2007 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
verisignroot	CN=VeriSign Universal Root Certification Authority, OU="(c) 2008 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
verisignserverclass3___g5	CN=VeriSign Class 3 Public Primary Certification Authority - G5, OU="(c) 2006 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	32:F3:08:82:62:2B:87:CF:88:56:C6:3D:B8:73:DF:08:53:B4:DD:27
workplaceca	CN=mySAP.com Workplace CA (dsa), O=mySAP.com Workplace, C=DE	A1:7D:8B:51:8A:8F:BB:DE:A5:00:C8:1E:96:12:26:16:32:4A:34:C0
xrampglobalca [jdk]	CN=XRamp Global Certification Authority, O=XRamp Security Services Inc, OU=www.xrampsecurity.com, C=US	B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6

Related Information

Server Certificate Authentication [page 93]
[Deprecated] Use Certificates Signed by a Trusted Certificate Authority [page 275]

7.4.2.3 Using the Keystore Service for Client Side HTTPS Connections

Prerequisites

- You have imported the Hello World sample in the Neo SDK. For more information, see Using Samples [page 850].
- You have an HTTPS server hosting a resource which you would like to access in your application.
- You have prepared the required key material as *.jks* files in the local file system.

> **i Note**
>
> File `client.jks` contains a client identity key pair trusted by the HTTPS server, and `cacerts.jks` contains all issuer certificates for the HTTPS server. The files are created with the *keytool* from the standard JDK distribution. For more information, see Key and Certificate Management Tool .

Context

This procedure describes how to extend the HelloWorld Web application to use Keystore Service. It includes making an SSL connection to an external HTTPS server by using the JDK and Apache HTTP Client.

You test and run the application on your local server and on SAP BTP.

Using javax.net.ssl Framework

Procedure

1. Declare a Keystore Service Resource Reference.

   To enable the look-up of the Keystore Service through JNDI, you need to add a resource reference entry to the `web.xml` descriptor.

   a. In the application archive, open the `HelloWorld/WebContent/WEB-INF/web.xml` file, and insert the following content:

   ```
   <resource-ref>
       <res-ref-name>KeyStoreService</res-ref-name>
   ```

```
    <res-type>com.sap.cloud.crypto.keystore.api.KeyStoreService</res-type>
</resource-ref>
```

b. Save the file.

2. Create a new Servlet.

a. In a Servlet class, use Keystore service. For example, use the following source code:

```
package com.sap.cloud.sample.keystoreservice;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.security.KeyStore;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.net.ssl.KeyManager;
import javax.net.ssl.KeyManagerFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.sap.cloud.crypto.keystore.api.KeyStoreService;
public class SSLExampleServlet extends HttpServlet {
   private static final long serialVersionUID = 1L;
   /**
    * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
    */
   protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
     // get Keystore Service
     KeyStoreService keystoreService;
     try {
       Context context = new InitialContext();
       keystoreService = (KeyStoreService) context.lookup("java:comp/env/
KeyStoreService");
     } catch (NamingException e) {
       response.getWriter().println("Error:<br><pre>");
       e.printStackTrace(response.getWriter());
       response.getWriter().println("</pre>");
       throw new ServletException(e);
     }
     String host = request.getParameter("host");
     if (host == null || (host = host.trim()).isEmpty()) {
       response.getWriter().println("Host is not specified");
       return;
     }
     String port = request.getParameter("port");
     if (port == null || (port = port.trim()).isEmpty()) {
       port = "443";
     }
     String path = request.getParameter("path");
     if (path == null || (path = path.trim()).isEmpty()) {
       path = "/";
     }
     String clientKeystoreName = "client";
```

```
     String clientKeystorePassword =
request.getParameter("client.keystore.password");
     if (clientKeystorePassword == null || (clientKeystorePassword =
clientKeystorePassword.trim()).isEmpty()) {
       response.getWriter().println("Password for client keystore is not
specified");
       return;
     }
     String trustedCAKeystoreName = "cacerts";
    // get a named keystores with password for integrity check
     KeyStore clientKeystore;
     try {
       clientKeystore = keystoreService.getKeyStore(clientKeystoreName,
clientKeystorePassword.toCharArray());
     } catch (Exception e) {
       response.getWriter().println("Client keystore is not available: " +
e);
       return;
     }
     // get a named keystore without integrity check
     KeyStore trustedCAKeystore;
     try {
       trustedCAKeystore =
keystoreService.getKeyStore(trustedCAKeystoreName, null);
     } catch (Exception e) {
       response.getWriter().println("Trusted CAs keystore is not
available" + e);
       return;
     }

  callHTTPSServer(response, host, port, path, clientKeystorePassword,
clientKeystore, trustedCAKeystore);
}
 private void callHTTPSServer(HttpServletResponse response,
                              String host,
                              String port,
                              String path,
                              String clientKeystorePassword,
                              KeyStore clientKeystore,
                              KeyStore trustedCAKeystore) throws
IOException {
   try {
     KeyManagerFactory kmf =
KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
     kmf.init(clientKeystore, clientKeystorePassword.toCharArray());
     KeyManager[] keyManagers = kmf.getKeyManagers();
     TrustManagerFactory tmf =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
     tmf.init(trustedCAKeystore);
     TrustManager[] trustManagers = tmf.getTrustManagers();
     SSLContext sslContext = SSLContext.getInstance("TLS");
     sslContext.init(keyManagers, trustManagers, null);
     SSLSocketFactory factory = sslContext.getSocketFactory();
     SSLSocket socket = (SSLSocket)factory.createSocket(host,
Integer.parseInt(port));
     socket.startHandshake();
     PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())));
     out.println("GET " + path + " HTTP/1.0");
     out.println();
     out.flush();
     if (out.checkError()) {
       response.getWriter().println("Error durring request sending");
     }
     BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
     String inputLine;
     while ((inputLine = in.readLine()) != null) {
```

```
        response.getWriter().println(inputLine);
      }
      in.close();
      out.close();
      socket.close();
    } catch (Exception e) {
      response.getWriter().println("Error:<br><pre>");
      e.printStackTrace(response.getWriter());
      response.getWriter().println("</pre>");
    } finally {
      response.getWriter();
    }
  }
}
```

b. Save the Java code and make sure that the project compiles without errors.

3. Deploy and Test the Web Application
   - Local Server Configuration of the Keystore
   - Cloud Configuration of the Keystore

Using Apache HTTP Client

Procedure

1. Add the required *.jar* files of the Apache HTTP Client (version 4.2 or higher) to the build path of your project.
2. Add the following imports:

```
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.scheme.SchemeSocketFactory;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
```

3. Replace `callHTTPSServer()` method with the one using Apache HTTP client.

```
private void callHTTPSServer(HttpServletResponse response,
                             String host,
                             String port,
                             String path,
                             String clientKeystorePassword,
                             KeyStore clientKeystore,
                             KeyStore trustedCAKeystore) throws IOException,
ServletException {
  try {
    SchemeSocketFactory socketFactory = new SSLSocketFactory(clientKeystore,
clientKeystorePassword, trustedCAKeystore);
    Scheme sch = new Scheme("https", Integer.parseInt(port), socketFactory);
    DefaultHttpClient httpclient = new DefaultHttpClient();
    httpclient.getConnectionManager().getSchemeRegistry().register(sch);
    HttpGet httpget = new HttpGet("https://" + host + path);
    HttpResponse resp = httpclient.execute(httpget);
    HttpEntity entity = resp.getEntity();
    BufferedReader in = new BufferedReader(new
InputStreamReader(entity.getContent()));
```

```
    String inputLine;
    while ((inputLine = in.readLine()) != null) {
      response.getWriter().println(inputLine);
    }
    EntityUtils.consume(entity);
  } catch (Exception e) {
    response.getWriter().println("error: " + e);
    throw new ServletException(e);
  } finally {
    response.getWriter().flush();
  }
}
```

Related Information

Configure Keystore on Local Server [page 1795]
Configure Keystore on Cloud [page 1796]

7.4.2.3.1 Configure Keystore on Local Server

Context

Procedure

1. To deploy your Web application on the local server, follow the steps for deploying a Web application locally as described in .
2. To upload the required keystores, copy the prepared `client.jks` and `cacerts.jks` files into `<local server root>\config_master\com.sap.cloud.crypto.keystore` subfolder.
3. To test the functionality, open the following URL in your Web browser: `http://localhost:<local server HTTP port>/HelloWorld/SSLExampleServlet?host=<remote HTTPS server host name>&port=<remote HTTPS server port number>&path=<remote HTTPS server resource>&client.keystore.password=<client identity keystore password>`.

Related Information

7.4.2.3.2 Configure Keystore on Cloud

Context

Procedure

1. To deploy your Web application on the cloud, follow the steps for deploying a Web application to SAP BTP as described in Deploy on the Cloud with the Console Client [page 891].
2. To upload the required keystores, execute `upload-keystore` console command with the prepared *.jks* files. For more information, see the *Cloud Configuration* section in Keys and Certificates [page 1781].

   > ### Example
   >
   > Assuming you have *mySubaccount* subaccount, *myApplication* application, *myUser* user, and the keystore files in folder `C:\Keystores`, you need to execute the following commands in your local `<SDK root>\tools` folder:
   >
   > ```
   > neo upload-keystore --account mySubaccount --user myUser --location C:
   > \Keystores\client.jks --host hana.ondemand.com
   > neo upload-keystore --account mySubaccount --user myUser --location C:
   > \Keystores\cacerts.jks
   >  --host hana.ondemand.com
   > ```
   >
   > For more information about the keystore console commands, see Keystore Console Commands [page 1783].

3. To test the functionality, open the application URL shown by SAP BTP cockpit with the following options:`<SAP BTP Application URL>/SSLExampleServlet?host=<remote HTTPS server host name>&port=<remote HTTPS server port number>&path=<remote HTTPS server resource>& client.keystore.password=<client identity keystore password>`.

   For more information, see Start and Stop Applications [page 1602].

Related Information

Deploy on the Cloud with the Console Client [page 891]
Keys and Certificates [page 1781]
Keystore Console Commands [page 1783]
Start and Stop Applications [page 1602]

7.4.3 Enabling Client Certificate Authentication

You can enable the users for your Web application to authenticate using client certificates. This corresponds to the CERT and BASICCERT authentication methods supported in Java EE.

Overview

Prerequisites

(For the mapping modes requiring certificate authorities) You have a keystore defined. See Keys and Certificates [page 1781].

Context

Using information in the client certificate, SAP BTP will map the certificate to a user name using the mapping mode you specify.

1. Configure Reverse Proxy to Request a Client Certificate

Context

By default, SAP BTP supports SSL communication for Web applications through a reverse proxy that does not request a client certificate. To enable client certificate authentication, you need to configure the reverse proxy to request a client certificate.

Add `cert.hana.ondemand.com` as a platform domain. See Using Platform Domains [page 1666].

For more information about the trusted certificate authorities (CAs) for SAP BTP, see Trusted Certificate Authorities for Client Certificate Authentication [page 1803].

2. Protect Application Resources

In your Web application, use declarative or programmatic authentication to protect application resources.

Use one of the following two methods for client certificate authentication:

- `CERT` - Users can authenticate only with client certificate.
- `BASICCERT` - Users can authenticate either with client certificate or with user name and password.

If you use the declarative approach, you need to specify the authentication method in the application web.xml file. See Declarative Authentication [page 1679].

If you use the programmatic approach, specify the authentication method as a parameter for the login context creation. For more information, see Programmatic Authentication [page 1683].

3. Define User Mapping

The user mapping defines how the user name is derived from the received client certificate. You configure user mapping using Java system properties.

Use the following system properties to define user mapping:

System Properties for User Mappings

System Property	Description
`com.sap.cloud.crypto.clientcert.mapping_mode`	(Mandatory) Defines how the received client certificate is interpreted.
`com.sap.cloud.crypto.clientcert.keystore_name`	Defines the name of the keystore used during the user mapping process, and it is mandatory for the mapping modes that use the keystore. **i Note** Use a keystore that is available in the Keystore Service. See Keys and Certificates [page 1781]. **i Note** Use the keystore name **without** the keystore file extension (jks for example). **i Note** Depending on the value of the `com.sap.cloud.crypto.clientcert.mapping_mode` property,using the `com.sap.cloud.crypto.clientcert.keystore_name` property may be mandatory.

For more information how to set the value of the system property, see Configure VM Arguments [page 1598].

For more information about the particular values you need to set, see the table below.

Mapping Mode	Description	How to Set	Example
CN	The user name equals the common name (CN) of the certificate's subject.	Set the `com.sap.cloud.crypto.clientcert.mapping_mode` property with value CN. In addition, if you want to accept certificates from trusted certificate authorities (CAs) only, set the `com.sap.cloud.crypto.clientcert.keystore_name` with a value the keystore name containing the trusted issuers. **i Note** The client certificate is not accepted if its issuer is not in the keystore or is not in a chain trusted by this keystore, and then the authentication fails. For more information about the Keystore Service, see Keys and Certificates [page 1781]. If you want to accept certificates from any issuer, skip setting the `com.sap.cloud.crypto.clientcert.keystore_name` property.	A client certificate with *cn=myuser,ou=security* as a subject is mapped to a *myuser* user name.

Mapping Mode	Description	How to Set	Example
CN@issuer	For this mapping mode, the user name is defined as *<CN of the certificate's subject>@<keystore alias of the certificate's issuer>*. Use this mapping mode when you have certificates with identical CNs.	To use this mapping mode, you have to set the following system properties: • `com.sap.cloud.crypto.clientcert.mapping_mode` with a value *CN@Issuer* • `com.sap.cloud.crypto.clientcert.keystore_name` with a value the keystore name containing the trusted issuers The issuer is trusted if it is in the keystore or is part of a trusted certificate chain. A certificate chain is trusted if at least one of its issuers exists in the keystore. **i Note** The client certificate is not accepted if its issuer is not in the keystore or is not in a chain trusted by this keystore, and then the authentication fails. For more information about setting the Keystore Service, see Keys and Certificates [page 1781].	A client certificate with *CN=john, C=DE, O=SAP, OU=Development* as a subject and *CN=SSO CA, O=SAP* as an issuer is received. The specified keystore with trusted issuers contains the same issuer, *CN=SSO CA, O=SAP*, that has an *sso_ca* alias. Then the user name is defined as *john@sso_ca*.

Mapping Mode	Description	How to Set	Example
wholeCert	For this mapping mode, the whole client certificate is compared with each entry in the specified keystore, and then the user name is defined as the alias of the matching entry.	To use this mapping mode, you have to set the following system properties: • `com.sap.cloud.crypto.clientcert.mapping_mode` with a value *wholeCert* • `com.sap.cloud.crypto.clientcert.keystore_name` with a value the keystore name containing the respective user certificates **Note** The client certificate is not accepted if no exact match is found in the specified keystore, and then the authentication fails. For more information about the Keystore Service, see Keys and Certificates [page 1781].	The following client certificate is received: `Subject:` *CN=john.miller, C=DE, O=SAP, OU=Development* `Validity Start Date:` *March 19 09:04:32 2013 GMT* `Validity End Date:` *March 19 09:04:32 2018 GMT* ... The specified keystore contains the same certificate with an alias *john*. Then the user name is defined as *john*.

Mapping Mode	Description	How to Set	Example
subjectAndIssuer	For this mapping mode, only the subject and issuer fields of the received client certificate are compared with the ones of each keystore entry, and then the user name is defined as the alias of the matching entry. Use this mapping mode when you want authentication by validating only the certificate's subject and issuer.	To use this mapping mode, you have to set the following system properties: • `com.sap.cloud.crypto.clientcert.mapping_mode` with a value *subjectAndIssuer* • `com.sap.cloud.crypto.clientcert.keystore_name` with a value the keystore name containing the respective user certificates **i Note** The client certificate is not accepted if an entry with the same subject and issuer is missing in the specified keystore, and then the authentication fails. For more information about the Keystore Service, see Keys and Certificates [page 1781].	A certificate with *CN=john.miller, C=DE, O=SAP, OU=Development* as a subject and *CN=SSO CA, O=SAP* as an issuer is received. The specified keystore contains a certificate with alias *john* that has the same subject and issuer fields. Then the user name is defined as *john*.

4. Call the Application Using Client Certificate Authentication

Context

After you set up client certificate authentication, you need to use a special URL to call the application with that authentication type. You need to use the following URL pattern:

`https://<my application>.cert.<SAP BTP region host>/<application resource>`

i Note

In comparison, the default application URL pattern at SAP BTP is:

`https://<my application>.<SAP BTP region host>/<application resource>`

So the difference is only in the **`.cert`** part.

For the available SAP BTP regions and their hosts, see Regions.

Example 1: You have an application running in the Europe (Rot) region. It has the following default application URL:

```
https://bigideaX.hana.ondemand.com/exampleX
```

To call the application with client certificate authentication, you need to use the following URL:

```
https://bigideaX.cert.hana.ondemand.com/exampleX
```

Example 2: You have an application running in the Canada (Toronto) region. It has the following default application URL:

```
https://bigideaZ.ca1.hana.ondemand.com/exampleZ
```

To call the application with client certificate authentication, you need to use the following URL:

```
https://bigideaZ.cert.ca1.hana.ondemand.com/exampleZ
```

7.4.3.1 Trusted Certificate Authorities for Client Certificate Authentication

To enable client certificate authentication in your application, users need to present client certificates issued by some of the certificate authorities (CAs) listed below.

Trusted CAs

Subject DN	Issuer DN	SHA1
CN=AddTrust External CA Root, OU=AddTrust External TTP Network, O=AddTrust AB, C=SE	CN=AddTrust External CA Root, OU=AddTrust External TTP Network, O=AddTrust AB, C=SE	02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
CN=Amazon Root CA 1, O=Amazon, C=US	CN=Amazon Root CA 1, O=Amazon, C=US	8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
CN=Baltimore CyberTrust Root, OU=CyberTrust, O=Baltimore, C=IE	CN=Baltimore CyberTrust Root, OU=CyberTrust, O=Baltimore, C=IE	D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
CN=Certum CA, O=Unizeto Sp. z o.o., C=PL	CN=Certum CA, O=Unizeto Sp. z o.o., C=PL	62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
CN=COMODO Certification Authority, O=COMODO CA Limited, L=Salford, ST=Greater Manchester, C=GB	CN=COMODO Certification Authority, O=COMODO CA Limited, L=Salford, ST=Greater Manchester, C=GB	EE:86:93:87:FF:FD:83:49:AB:5A:D1:43:22:58:87:89:A4:57:B0:12
CN=COMODO RSA Certification Authority, O=COMODO CA Limited, L=Salford, ST=Greater Manchester, C=GB	CN=COMODO RSA Certification Authority, O=COMODO CA Limited, L=Salford, ST=Greater Manchester, C=GB	AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4

Subject DN	Issuer DN	SHA1
CN=DigiCert Assured ID Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US	CN=DigiCert Assured ID Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US	05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
CN=DigiCert Global Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US	CN=DigiCert Global Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US	A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
CN=DigiCert Global Root G2, OU=www.digicert.com, O=DigiCert Inc, C=US	CN=DigiCert Global Root G2, OU=www.digicert.com, O=DigiCert Inc, C=US	DF:3C:24:F9:BF:D6:66:76:1B:26:80:73:FE:06:D1:CC:8D:4F:82:A4
CN=DigiCert High Assurance EV Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US	CN=DigiCert High Assurance EV Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US	5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
CN=DST Root CA X3, O=Digital Signature Trust Co.	CN=DST Root CA X3, O=Digital Signature Trust Co.	DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
CN=Entrust Root Certification Authority - G2, OU="(c) 2009 Entrust, Inc. - for authorized use only", OU=See www.entrust.net/legal-terms, O="Entrust, Inc.", C=US	CN=Entrust Root Certification Authority - G2, OU="(c) 2009 Entrust, Inc. - for authorized use only", OU=See www.entrust.net/legal-terms, O="Entrust, Inc.", C=US	8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
CN=Entrust Root Certification Authority, OU="(c) 2006 Entrust, Inc.", OU=www.entrust.net/CPS is incorporated by reference, O="Entrust, Inc.", C=US	CN=Entrust Root Certification Authority, OU="(c) 2006 Entrust, Inc.", OU=www.entrust.net/CPS is incorporated by reference, O="Entrust, Inc.", C=US	B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
CN=Entrust.net Certification Authority (2048), OU=(c) 1999 Entrust.net Limited, OU=www.entrust.net/CPS_2048 incorp. by ref. (limits liab.), O=Entrust.net	CN=Entrust.net Certification Authority (2048), OU=(c) 1999 Entrust.net Limited, OU=www.entrust.net/CPS_2048 incorp. by ref. (limits liab.), O=Entrust.net	50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
CN=GeoTrust Global CA, O=GeoTrust Inc., C=US	CN=GeoTrust Global CA, O=GeoTrust Inc., C=US	DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
CN=GeoTrust Primary Certification Authority - G2, OU=(c) 2007 GeoTrust Inc. - For authorized use only, O=GeoTrust Inc., C=US	CN=GeoTrust Primary Certification Authority - G2, OU=(c) 2007 GeoTrust Inc. - For authorized use only, O=GeoTrust Inc., C=US	8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
CN=GeoTrust Primary Certification Authority - G3, OU=(c) 2008 GeoTrust Inc. - For authorized use only, O=GeoTrust Inc., C=US	CN=GeoTrust Primary Certification Authority - G3, OU=(c) 2008 GeoTrust Inc. - For authorized use only, O=GeoTrust Inc., C=US	03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
CN=GeoTrust Primary Certification Authority, O=GeoTrust Inc., C=US	CN=GeoTrust Primary Certification Authority, O=GeoTrust Inc., C=US	32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
CN=GeoTrust Universal CA, O=GeoTrust Inc., C=US	CN=GeoTrust Universal CA, O=GeoTrust Inc., C=US	E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
CN=GlobalSign Root CA, OU=Root CA, O=GlobalSign nv-sa, C=BE	CN=GlobalSign Root CA, OU=Root CA, O=GlobalSign nv-sa, C=BE	B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C

Subject DN	Issuer DN	SHA1
CN=GlobalSign, O=GlobalSign, OU=GlobalSign Root CA - R2	CN=GlobalSign, O=GlobalSign, OU=GlobalSign Root CA - R2	75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
CN=GlobalSign, O=GlobalSign, OU=GlobalSign Root CA - R3	CN=GlobalSign, O=GlobalSign, OU=GlobalSign Root CA - R3	D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
CN=Go Daddy Root Certificate Authority - G2, O="GoDaddy.com, Inc.", L=Scottsdale, ST=Arizona, C=US	CN=Go Daddy Root Certificate Authority - G2, O="GoDaddy.com, Inc.", L=Scottsdale, ST=Arizona, C=US	47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
CN=QuoVadis Root CA 2 G3, O=QuoVadis Limited, C=BM	CN=QuoVadis Root CA 2 G3, O=QuoVadis Limited, C=BM	09:3C:61:F3:8B:8B:DC:7D:55:DF:75:38:02:05:00:E1:25:F5:C8:36
CN=QuoVadis Root CA 2, O=QuoVadis Limited, C=BM	CN=QuoVadis Root CA 2, O=QuoVadis Limited, C=BM	CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
CN=SAP Cloud Root CA 01, O=SAP SE, C=DE	CN=SAP Cloud Root CA 01, O=SAP SE, C=DE	05:A0:64:F7:16:E3:6C:AE:5A:BB:DD:E2:17:42:72:56:EA:D8:B4:A7
CN=SAP Cloud Root CA, O=SAP SE, L=Walldorf, C=DE	CN=SAP Cloud Root CA, O=SAP SE, L=Walldorf, C=DE	6D:80:92:77:4A:F2:D5:ED:AE:3A:5C:99:D6:56:93:1C:21:97:A9:50
CN=SAP Global Root CA, O=SAP AG, L=Walldorf, C=DE	CN=SAP Global Root CA, O=SAP AG, L=Walldorf, C=DE	0A:B6:2A:F4:7F:E5:59:84:7D:79:8A:1F:C4:E1:7F:67:FD:7E:82:4C
CN=SAP Internet of Things CA, O=SAP IoT Trust Community II, C=DE	CN=SAP Internet of Things CA, O=SAP IoT Trust Community II, C=DE	45:53:D3:F2:22:58:FE:35:59:B1:84:9F:27:3B:8C:69:C2:4C:FA:15
CN=SAP Passport CA, O=SAP Trust Community, C=DE	CN=SAP Passport CA, O=SAP Trust Community, C=DE	10:BD:99:32:E8:3A:01:CD:C4:4F:56:10:05:47:30:A8:73:18:16:6D
CN=SSO_CA, O=SAP-AG, C=DE	CN=SSO_CA, O=SAP-AG, C=DE	4D:11:61:08:30:D7:B3:1C:62:87:19:8E:95:D5:5F:3E:8F:05:E4:0B
CN=Starfield Services Root Certificate Authority - G2, O="Starfield Technologies, Inc.", L=Scottsdale, ST=Arizona, C=US	CN=Starfield Services Root Certificate Authority - G2, O="Starfield Technologies, Inc.", L=Scottsdale, ST=Arizona, C=US	92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
CN=SwissSign Gold CA - G2, O=SwissSign AG, C=CH	CN=SwissSign Gold CA - G2, O=SwissSign AG, C=CH	D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
CN=SwissSign Platinum CA - G2, O=SwissSign AG, C=CH	CN=SwissSign Platinum CA - G2, O=SwissSign AG, C=CH	56:E0:FA:C0:3B:8F:18:23:55:18:E5:D3:11:CA:E8:C2:43:31:AB:66
CN=SwissSign Silver CA - G2, O=SwissSign AG, C=CH	CN=SwissSign Silver CA - G2, O=SwissSign AG, C=CH	9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
CN=T-TeleSec GlobalRoot Class 2, OU=T-Systems Trust Center, O=T-Systems Enterprise Services GmbH, C=DE	CN=T-TeleSec GlobalRoot Class 2, OU=T-Systems Trust Center, O=T-Systems Enterprise Services GmbH, C=DE	59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
CN=thawte Primary Root CA - G2, OU="(c) 2007 thawte, Inc. - For authorized use only", O="thawte, Inc.", C=US	CN=thawte Primary Root CA - G2, OU="(c) 2007 thawte, Inc. - For authorized use only", O="thawte, Inc.", C=US	AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12

Subject DN	Issuer DN	SHA1
CN=thawte Primary Root CA - G3, OU="(c) 2008 thawte, Inc. - For authorized use only", OU=Certification Services Division, O="thawte, Inc.", C=US	CN=thawte Primary Root CA - G3, OU="(c) 2008 thawte, Inc. - For authorized use only", OU=Certification Services Division, O="thawte, Inc.", C=US	F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
CN=thawte Primary Root CA, OU="(c) 2006 thawte, Inc. - For authorized use only", OU=Certification Services Division, O="thawte, Inc.", C=US	CN=thawte Primary Root CA, OU="(c) 2006 thawte, Inc. - For authorized use only", OU=Certification Services Division, O="thawte, Inc.", C=US	91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
CN=USERTrust RSA Certification Authority, O=The USERTRUST Network, L=Jersey City, ST=New Jersey, C=US	CN=USERTrust RSA Certification Authority, O=The USERTRUST Network, L=Jersey City, ST=New Jersey, C=US	2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
CN=VeriSign Class 3 Public Primary Certification Authority - G3, OU="(c) 1999 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	CN=VeriSign Class 3 Public Primary Certification Authority - G3, OU="(c) 1999 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
CN=VeriSign Class 3 Public Primary Certification Authority - G4, OU="(c) 2007 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	CN=VeriSign Class 3 Public Primary Certification Authority - G4, OU="(c) 2007 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
CN=VeriSign Class 3 Public Primary Certification Authority - G5, OU="(c) 2006 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	CN=VeriSign Class 3 Public Primary Certification Authority - G5, OU="(c) 2006 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
CN=VeriSign Universal Root Certification Authority, OU="(c) 2008 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	CN=VeriSign Universal Root Certification Authority, OU="(c) 2008 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US	36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
OU=Go Daddy Class 2 Certification Authority, O="The Go Daddy Group, Inc.", C=US	OU=Go Daddy Class 2 Certification Authority, O="The Go Daddy Group, Inc.", C=US	27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
OU=Starfield Class 2 Certification Authority, O="Starfield Technologies, Inc.", C=US	OU=Starfield Class 2 Certification Authority, O="Starfield Technologies, Inc.", C=US	AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A

For a complete list of Root CA certificates that are approved by SAP Global Security, see SAP Note 2801396.

7.4.4 Enable Strong Encryption in Applications

This is a deprecated procedure how to install Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files on SAP JVM to enable unlimited cryptography power. You do not need to use it anymore. All supported runtimes already come with a Java version that supports strong encryption.

Prerequisites

You have the appropriate Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files enabling cryptography with unlimited strength.

Context

Procedure

1. Pack the encryption policy files (JCE Unlimited Strength Jurisdiction Policy Files) in the following folder of the Web application:

   `META-INF/ext_security/jre7` - for applications, running on JDK 1.7
2. If the application consists of multiple WAR files, pack the encryption policy files in one of them.
3. Deploy the application on SAP BTP.

Results

The encryption policy files (Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files) will be installed on the JVM of the application prior to start. As a result, the application can use unlimited strength encryption.

Example

The WAR file of the application must have the following file entries:

```
META-INF/ext_security/jre7/local_policy.jar
```

```
META-INF/ext_security/jre7/US_export_policy.jar
```

Related Information

Deploying and Updating Applications [page 875]

7.4.5 Storing Passwords

Context

Using the Password Storage API, you can securely persist passwords and key phrases such as passwords for keystore files. Once persisted in the password storage, they:

- Can be accessed from different application computing units;
- Survive application restarts and updates;
- Are a subject of automatic backup;
- Stay persisted unless you explicitly delete them via the API, or you undeploy your application.

Before transportation and persistence, passwords are encrypted with an encryption key which is specific for the application that owns the password. They are stored according to subscription, and accessible only when the owning application is working on behalf of the corresponding subscription.

> **i Note**
>
> Each password is identified by an alias. To check the rules and constraints about passwords aliases, permitted characters and length, see the security javadoc.

Adding Resource Reference

To use the password storage API, you need to add a resource reference to `PasswordStorage` in the `web.xml` file of your application, which is located in the `\WebContent\WEB-INF` folder as shown below:

```
<resource-ref>
  <res-ref-name>PasswordStorage</res-ref-name>
  <res-type>com.sap.cloud.security.password.PasswordStorage</res-type>
</resource-ref>
```

Looking Up the JNDI

An initial JNDI context can be obtained by creating a `javax.naming.InitialContext` object. You can then consume the resource by looking up the naming environment through the `InitialContext` class as follows:

```
InitialContext ctx = new InitialContext();
PasswordStorage passwordStorage = (PasswordStorage) ctx.lookup("java:comp/env/
PasswordStorage");
```

Note that according to the Java EE Specification, the prefix **java:comp/env** should be added to the JNDI resource name (as specified in the `web.xml` file) to form the lookup name.

Using the Password Storage API

Below is a code example of how to use the API to set, get or delete passwords. These methods provide the option of assigning an alias to the password.

```
import javax.naming.InitialContext;
import javax.naming.NamingException;

import com.sap.cloud.security.password.PasswordStorage;
import com.sap.cloud.security.password.PasswordStorageException;
.......

   private PasswordStorage getPasswordStorage() throws NamingException {
    InitialContext ctx = new InitialContext();
    PasswordStorage passwordStorage = (PasswordStorage)
ctx.lookup("java:comp/env/PasswordStorage");
    return passwordStorage;
  }

  private void setPassword(String alias, char[] password) throws
PasswordStorageException, NamingException {
    PasswordStorage passwordStorage = getPasswordStorage();
    passwordStorage.setPassword(alias, password);
  }

  private char[] getPassword(String alias) throws PasswordStorageException,
NamingException {
    PasswordStorage passwordStorage = getPasswordStorage();
    return passwordStorage.getPassword(alias);
  }

  private void deletePassword(String alias) throws PasswordStorageException,
NamingException {
    PasswordStorage passwordStorage = getPasswordStorage();
    return passwordStorage.deletePassword(alias);
  }
```

> **i** Note
>
> It is recommended to cache the obtained value, as reading of passwords is an expensive operation and involves several internal remote calls to central storage and audit infrastructure. As passwords are different for the different tenant the cache should be tenant aware. PasswordsStorage instance obtained via lookup can be cached and used by multiple threads.

Local Testing

When you run applications on SAP BTP local runtime, you can use a local implementation of the password storage API, but keep in mind that the passwords are not encrypted and stored in a local file. Therefore, for local testing, use only test passwords.

Related Information

Security Development [page 1677]

7.5 Audit Logging in the Neo Environment

In this section you can find information for audit log functionalities in the SAP BTP Neo environment.

Parent topic: Security, Neo Environment [page 1671]

Related Information

Authorization and Trust Management in the Neo Environment [page 1672]
Platform Identity Provider [page 1746]
OAuth 2.0 Service [page 1751]
Keystore Service [page 1779]
Principal Propagation [page 1816]
Protection from Web Attacks [page 1835]
Data Protection and Privacy [page 1848]
Audit Log Retrieval API Usage for the Neo Environment [page 1810]
Audit Log Retention API Usage for the Neo Environment [page 1815]

7.5.1 Audit Log Retrieval API Usage for the Neo Environment

The audit log retrieval API allows you to retrieve the audit logs for your SAP BTP Neo environment account. It follows the OData 4.0 standard, providing the audit log results as OData with collection of JSON entities.

The audit log retrieval API is protected with OAuth 2.0 client credentials.

It provides two OAuth scopes:

- Read Audit Logs – allow usage of the Audit Log Retrieval API to retrieve audit logs;
- Manage Audit Logs – allow usage of the Audit Log Retention API to read retention and set custom retention.

To call the API methods, create an OAuth client and obtain an access token. See Using Platform APIs [page 1149].

Example: Get audit logs for the last 30 days

To achieve this you need to execute a similar GET request:

```
https://api.<region host>/auditlog/v1/accounts/<account>/AuditLogRecords?
$count=true
```

> **i Note**
>
> The account provided as part of the URL should be the randomly generated technical name of the subaccount not the Display Name.

To authenticate, in the header provide the taken OAuth token similar to: "`Authorization: Bearer 41fce723412c6c18961f7e95d911ad37`"

The returned results would be split on pages with size – the default server page size. If the number of results is higher than the default server page size, in the response *@odata.nextLink* would be provided with the URL, to retrieve the next results' chunk.

Example: Get audit log filtering by time, user, category and application

To do this you need to execute a similar GET request:

```
https://api.<region host>/auditlog/v1/accounts/<account>/AuditLogRecords?
$filter=(Time le '2017-12-30T17.13.22' or Time eq '2017-12-30T17.13.22') and
User eq '<user>' and Category eq '<category>' and Application eq
'<application_name>'
```

> **i Note**
>
> In the above example you can use all of the OData v4.0 supported logical operators for filtering described in the following documentation, see Query Options .

> **i Note**
>
> You can only filter by time, user, category and application. You can combine multiple filters in one request.

> **i Note**
>
> When filtering by time the format should be: *yyyy-mm-ddThh.MM.ss*

To authenticate, in the header provide the taken OAuth token similar to: "`Authorization: Bearer 41fce723412c6c18961f7e95d911ad37`"

The returned results would be split on pages with size – the default server page size. If the number of results is higher than the default server page size, in the response *@odata.nextLink* would be provided with the URL, to retrieve the next results' chunk.

The predefined audit log message categories are:

- Data protection and privacy related

- *audit.data-access* read-access logging records for access to sensitive personal data;
- *audit.data-modification* data modification logging records for sensitive personal data.

Security related

- *audit.security-events* logging of general security events like login, logout, and other.;
- *audit.configuration* logging of security critical configuration changes.

For more infromation, see Change Logging and Read-Access Logging [page 1851].

Example: Get audit logs using client-side pagination

To get results based on pages with size 50, first check the total results number, execute a similar GET request:

```
https://api.<region host>/auditlog/v1/accounts/<account>/AuditLogRecords?
$count=true
```

To split the pages on a desired size, 50 results per page in this example, execute a similar GET request:

```
https://api.<region host>/auditlog/v1/accounts/<account>/AuditLogRecords?$top=50&
$skip=0
```

To get a second page, execute a similar GET request:

```
https://api.<region host>/auditlog/v1/accounts/<account>/AuditLogRecords?$top=50&
$skip=50
```

To get a third page, execute a similar GET request:

```
https://api.<region host>/auditlog/v1/accounts/<account>/AuditLogRecords?$top=50&
$skip=100
```

Continue the same requesting pattern, until the number of the results returned by count, in the first example of this section, is reached.

To authenticate, in the header provide the taken OAuth token similar to: "`Authorization: Bearer 41fce723412c6c18961f7e95d911ad37`"

> **i Note**
>
> If you use client-side pagination and request a client-side page bigger that the server-side default page, the audit log retrieval API will split the requested page in several chunks that would be returned. As a result you will receive a response containing an @odata.nextLink field, where the next data chunk could be retrieved (for more information, see the *Results* section below). Go to the next client-side page value only after you have iterated all the chunks the server breaks the result to, which means that there is no @odata.nextLink field as part of the response provided.

Results

Executing a GET request towards the audit log retrieval API, results in response similar to the one below. The information for the AuditLogRecords can be checked in the metadata OData part. In the "*value*" part you

receive the audit log messages in the format shown in the response example. The results returned on page are limited to the server page size. To get the next result page, navigate to the URL provided in *@odata.nextLink*.

Sample Code

```
{
                    "@odata.context": "$metadata#AuditLogRecords",
                    "value": [
                    {
                    "Uuid": "3b8a8b-16247c70836-8",
                    "Category": "audit.data-access",
                    "User": "<user>",
                    "Tenant": "<tenant>",
                    "Account": "<account>",
                    "Application": "<application>",
                    "Time": "2018-03-21T09.00.40.572+0000",
                    "Message": "Read data access message. \"%void\"The
accessed data belongs to {\"type\":\"account\",\"role\":\"account\",\"id\":
{\"id :\":\"auditlog\"}} and  read  from object with name \"Auditlog
Retrieval API\" and identifier {\"type\":\"Legacy.Object\",\"id\":{\"key\":
\"Auditlog Retrieval API\"}} by user null",
                    "InstanceId": null,
                    "FormatVersion": "2.2"
                    },
                    …
                    {
                    "Uuid": "33a87d-1621e7debb2-1be",
                    "Category": "audit.security-events",
                    "User": "<user>",
                    "Tenant": "<tenant>",
                    "Account": "<account>",
                    "Application": "<application>",
                    "Time": "2018-03-21T09.00.40.782+0000",
                    "Message": "Security event message. Security event: "This
is my message with custom parameters: &param1, &param2\",\"param1\":\"value
of param1\",\"param2\":\"value of param2\"",
                    "InstanceId": null,
                    "FormatVersion": "2.2"
                    },    ],
                    "@odata.nextLink": "http://localhost:8001/auditlog/v1/
accounts/auditlog/AuditLogRecords?$top=5000&$skip=0&$skiptoken=1000"
                    }


                    //Second Page:
                    {
                    "@odata.context": "$metadata#AuditLogRecords",
                    "value": [
                    {
                    "Uuid": "2a70bd-1621a471259-3653",
                    "Category": "audit.configuration",
                    "User": "<user>",
                    "Tenant": "<tenant>",
                    "Account": "<account>",
                    "Application": "<application>",
                    "Time": "2018-03-20T15.59.14.878+0000",
                    "Message": "Configuration change message. Attribute
attributes update from value \"[old value]\" to value \"[new value]\". ",
                    "InstanceId": null,
                    "FormatVersion": "2.2"
                    },
                    …
                    {
                    "Uuid": "33a87d-1621e7debb2-1bf",
                    "Category": "audit.data-modification",
                    "User": "<user>",
                    "Tenant": "<tenant>",
```

```
                    "Account": "<account>",
                    "Application": "<application>",
                    "Time": "2018-03-20T15.59.14.898+0000",
                    "Message": "Data modification message. Attribute
attribute1 update from value \"some old value\" to value \"some new value\".
Attribute attribute3 update from value \"old Value\" to value \"new Value\".
The data  update  from object with name \"object1 display name\" and
identifier {\"type\":\"Legacy.Object\",\"id\":{\"key\":\"object1_ID\"}} by
user null",
                    "InstanceId": null,
                    "FormatVersion": "2.2"
                    },
                    ],
                    "@odata.nextLink": "http://localhost:8001/auditlog/v1/
accounts/auditlog/AuditLogRecords?$top=5000&$skip=0&$skiptoken=2000"
                    }
```

The retrieved audit logs are in JSON format. The semantics of the JSON fields are as follows:

UUID	Unique identifier of the audit log message
Category	Category of the audit log message. It could be one of the pre-defined audit log types (audit.security-events , audit.configuration , audit.data-access or audit.data-modification) or a subcategory provided when invoking the “log” method with “subcategory” parameter (e.g. audit.data-modification.test , audit.data-access.my-sub-category etc.)
User	The user that has executed the auditable event. The result of the user field could be: • “<userA>” - the user is determined implicitly by audit logging; • “<userA> on behalf of <userB>” – the user A is implicitly determined by audit logging and the user B is set by the component writing audit logs and verified by audit logging; • “on behalf of <userB>” - the user B is set by the component writing audit logs and verified by audit logging; • “” (empty string) – no user is determined or verified by audit logging. **i Note** Users that are set by the component writing audit logs and not further verified as a validity by audit logging are visible **only** in the “*Message*” field in the “*Custom defined attributes*” part in field “*caller_user*”.
Tenant	Tenant ID owner of the auditable event.
Account	Account ID owner of the auditable event.
Application	Application that has generated the audit log event.

UUID	Unique identifier of the audit log message
Time	Timestamp when the auditable event is generated.
Message	Audit log message text.
InstanceId	Instance ID where the event has occurred if applicable.
FormatVersion	Audit log message format.

Related Information

Audit Log Retrieval API for the Neo Environment
API Documentation [page 1147]

7.5.2 Audit Log Retention API Usage for the Neo Environment

The audit log retention API allows you to view your currently active retention period for all the audit log data that is stored for your account.

The audit log data stored for your account will be retained for 201 days, after which it will be deleted.

The length of the retention period comes from SAP specific requirements. Using the API, you can modify the default retention period for a customer retention period that corresponds to your legal, business, or other restrictions.

> **i Note**
>
> The setup of a custom retention for the first time is related to data migration that could last for up to 24 hours. The audit log retention API may return `inconsistency` as a results from the migration during that time frame. All the audit logs written during the transition period are stored and are not lost. They would be visible after the initial transition stage is over. This however is not valid for all the subsequent changes in the retention period made for the same account.

The audit log retention API is protected with OAuth 2.0 client credentials.

It provides two OAuth scopes:

- Read Audit Logs – allow usage of the Audit Log Retrieval API to retrieve audit logs;
- Manage Audit Logs – allow usage of the Audit Log Retention API to read retention and set custom retention.

Create an OAuth client and obtain an access token to call the API methods. See Using Platform APIs [page 1149].

Example: Get your currently active audit log retention period

To do this you have to execute a similar GET request:

```
https://api.<region host>/auditlog/v1/retention/accounts/<account>/
AuditLogRetention
```

To authenticate, in the header provide the taken OAuth token similar to: "`Authorization: Bearer 41fce723412c6c18961f7e95d911ad37`"

The returned results is in a similar JSON format:

```
{ retention : <retention in days> }
```

Example: Change your active retention period for the audit log data for your account

To do this change, you have to execute a similar POST request:

```
https://api.<region host>/auditlog/v1/retention/accounts/<account>/
AuditLogRetention
```

To authenticate, in header provide the taken OAuth token similar to: "`Authorization: Bearer 41fce723412c6c18961f7e95d911ad37`"

With the following contents in the body part:

```
{ retention : <retention in days> }
```

> **i Note**
>
> Usage of audit log's custom retention period currently does not incur additional charges. This may change in the future and a fee based on the stored data volume and retention period can be applied.

Related Information

Audit Log Retention API
API Documentation [page 1147]

7.6 Principal Propagation

Exchange user ID information between systems or environments in SAP BTP.

In This Section

- Principal Propagation from the Cloud Foundry to the Neo Environment [page 1827]
- Principal Propagation from the Neo to the Cloud Foundry Environment [page 1821]

Other Principal Propagation Scenarios

- On-Premise User Store [page 1736]
- Principal Propagation to OAuth-Protected Applications [page 1770]
- Connectivity in the Cloud Foundry Environment: Principal Propagation
- Connectivity in the Neo Environment: Principal Propagation

Parent topic: Security, Neo Environment [page 1671]

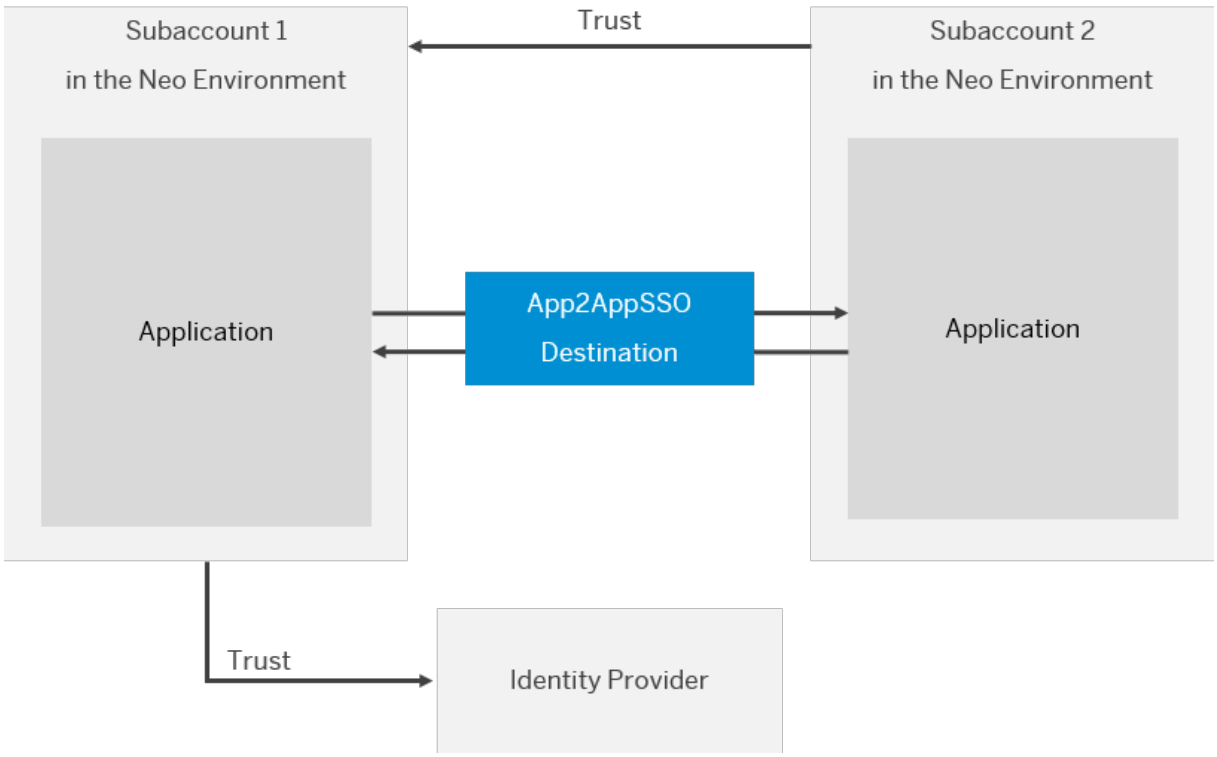
Related Information

Authorization and Trust Management in the Neo Environment [page 1672]
Platform Identity Provider [page 1746]
OAuth 2.0 Service [page 1751]
Keystore Service [page 1779]
Audit Logging in the Neo Environment [page 1810]
Protection from Web Attacks [page 1835]
Data Protection and Privacy [page 1848]
Propagate User Information Between Applications or Services

7.6.1 Principal Propagation Between Neo Subaccounts

Enable an application in your subaccount (Neo environment) to access an application in another account (Neo environment) without user login / user interaction in the second application. The second application propagates its logged-in user to the first application using an AppToAppSSO destination.

The graphic below illustrates the overall setup of the scenario.



Contents

- Create Trust Between the Subaccounts [page 1818]
- Create a Destination [page 1820]

7.6.1.1 Create Trust Between the Subaccounts

Prerequisites

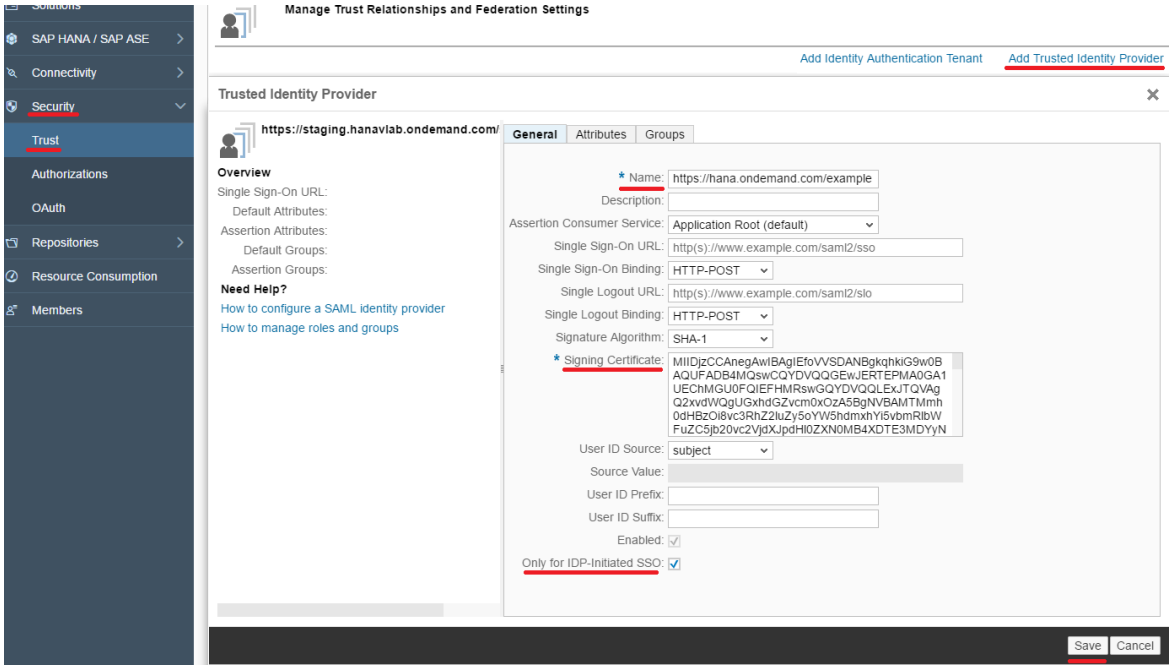
- You have an account with *Administrator* role in both SAP BTP subaccounts. See Managing Member Authorizations in the Neo Environment [page 1301].
- You have deployed both applications on SAP BTP. See Deploying and Updating Applications [page 875].
- You have a *custom* local service provider configuration in both subaccounts (this means in *cloud cockpit* > *Security* > *Trust* > *Local Service Provider* > you have chosen *Configuration Type* > *Custom*). See Configure the Local Service Provider [page 1721].

Procedure

1. Get the *Local Provider Name* and the *Signing Certificate* from the first subaccount.
   a. In SAP BTP cockpit, choose the first subaccount. See Navigate in the Cockpit.
   b. Navigate to *Security* > *Trust* > *Local Service Provider*.
   c. Save into a file the values of *Local Provider Name* and *Signing Certificate*.
   d. Make sure the value of *Principal Propagation* is set to *Enabled*.
2. Create trust on the second subaccount.
   a. In the SAP BTP cockpit, choose the second subaccount and navigate to the *Trust* tab.
   b. On the *Application Identity Provider* tab, choose *Add Trusted Identity Provider*. Provide the following information:

Field	Description
Name	The *Local Provider Name* of the first subaccount, which you copied in step 1.
Signing Certificate	The *Signing Certificate* of the first subaccount, which you copied in step 1.

   c. If it is not automatically checked, select the checkbox *Only for IDP-Initiated SSO*.
   d. Save the changes.



7.6.1.2 Create a Destination

Context

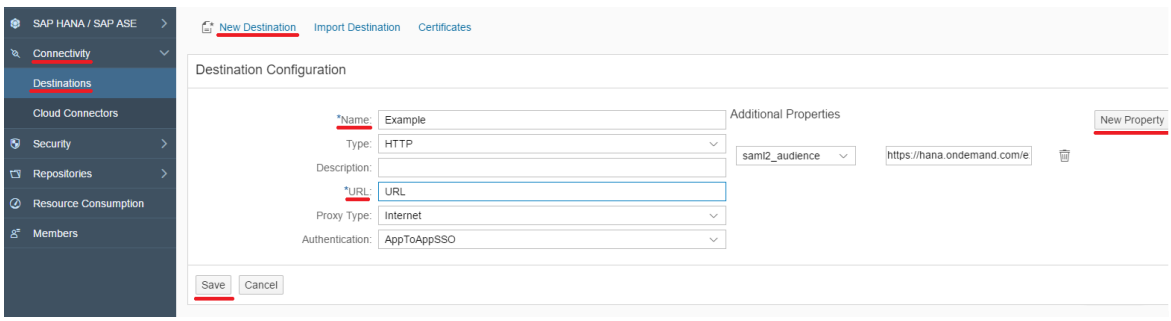
Connect the first subaccount, to the second subaccount by describing the source connection properties in a destination. For more information see Modeling Destinations [page 1072].

Procedure

1. Choose the first subaccount and navigate to *Connectivity* > *Destinations*.
2. Choose the *New Destination* tab.
3. Provide the following information:

Field	Description
Name	Technical name of the destination. It can be used later on to get an instance of that destination. It should be unique for the current application. **i Note** The name can contain only alphanumeric characters, underscores, and dashes. The maximum length is 200 characters.
URL	The URL of the protected resource that you want to access (the first application). See Configuring Application URLs [page 1644]. Example: `https://myappmysubaccount.hana.ondemand.com/`
Authentication	AppToAppSSO

4. Choose the *New Property* button. In the fields that appear, fill in: *saml2_audience* and enter the *Local Provider Name* of the second subaccount.
5. Save the changes.



6. Use the destination, as described in the Application-to-Application SSO Authentication [page 104] documentation.

Results

Using application-to-application communication you can now propagate the logged-in user of the second application.

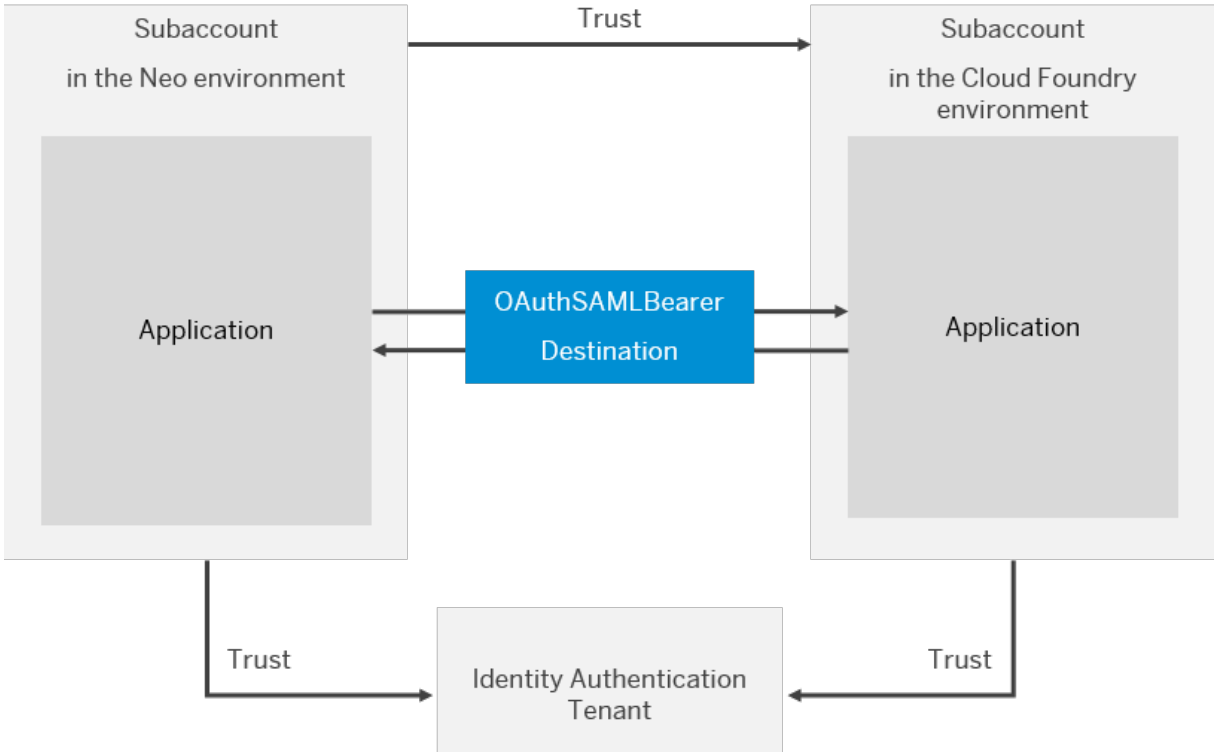
Related Information

Application-to-Application SSO Authentication [page 104]
Principal Propagation [page 129]

7.6.2 Principal Propagation from the Neo to the Cloud Foundry Environment

Enable an application in your subaccount in the Neo environment to access another application in a subaccount in the Cloud Foundry environment without user login (and user interaction) in the second application. For this scenario to work, the two subaccounts need to be in mutual trust, and in trust with the same Identity Authentication tenant. The second application will propagate its logged-in user to the first application using an OAuth2SAMLBearer destination.

The graphic below illustrates the overall setup of the scenario.



Prerequisites

- You have a user account with *Administrator* role in both SAP BTP subaccounts. See Managing Member Authorizations in the Neo Environment [page 1301].
- You have a custom local service provider configuration (signing keys and certificates, etc.) in your subaccount in the Neo environment. See Configure the Local Service Provider [page 1721].
- Both accounts have a trust configuration to the same Identity Authentication tenant. See:
  - Identity Authentication Tenant as an Application Identity Provider [page 1734] (for the Neo environment)
  - Manually Establish Trust and Federation Between UAA and Identity Authentication (for the Cloud Foundry environment)
- You have developed and deployed both applications, each in the corresponding subaccount.

> **i Note**
>
> All configuration steps described in this tutorial are done using the cloud cockpit.

Requirements to the Application in the Neo Environment

The application is running on a Java Web Tomcat 8 runtime.

In the source code, the application needs to reference the destination that we are about to create as a later step. The sample source code below illustrates a complete servlet working with the destination with name `pptest`.

```
package com.sap.cloud.samples;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;
import java.security.KeyStore;
import java.util.List;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManagerFactory;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.sap.core.connectivity.api.authentication.AuthenticationHeader;
import com.sap.core.connectivity.api.authentication.AuthenticationHeaderProvider;
import com.sap.core.connectivity.api.configuration.ConnectivityConfiguration;
import com.sap.core.connectivity.api.configuration.DestinationConfiguration;
@WebServlet("/neotocf")
public class NeoToCF extends HttpServlet {
  private static final long serialVersionUID = 1L;
  private static final Logger LOGGER = LoggerFactory.getLogger(NeoToCF.class);
  private static final String ON_PREMISE_PROXY = "OnPremise";
```

```
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
    response.getWriter().println("Served to: " +
request.getUserPrincipal().getName());
    try {
      // Look up the connectivity configuration API
      Context ctx = new InitialContext();
      ConnectivityConfiguration configuration = (ConnectivityConfiguration) ctx
                    .lookup("java:comp/env/connectivityConfiguration");
      // Get destination configuration
      DestinationConfiguration destConfiguration =
configuration.getConfiguration("pptest");
      if (destConfiguration == null) {
        response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                        String.format(
                                "Destination %s is not found. Hint:" +
                                       " Make sure to have the destination
configured.",
                                "pptest"));
                return;
      }

      AuthenticationHeaderProvider authHeaderProvider =
(AuthenticationHeaderProvider) ctx
                    .lookup("java:comp/env/myAuthHeaderProvider");
      // retrieve the authorization header for OAuth SAML Bearer principal
propagation
      List<AuthenticationHeader> samlBearerHeader = authHeaderProvider
                    .getOAuth2SAMLBearerAssertionHeaders(destConfiguration);
      LOGGER.debug("JWT token from CF XSUAA: " +
samlBearerHeader.get(1).getValue());

      // get the configured truststore
      KeyStore trustStore = destConfiguration.getTrustStore();

      // create sslcontext
      TrustManagerFactory tmf =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
      tmf.init(trustStore);

      SSLContext sslcontext = SSLContext.getInstance("TLS");
      sslcontext.init(null, tmf.getTrustManagers(), null);
      SSLSocketFactory sslSocketFactory = sslcontext.getSocketFactory();

      // get the destination URL
      String value = destConfiguration.getProperty("URL");
      URL url = new URL(value);

      // use the sslcontext for url connection
      URLConnection urlConnection = url.openConnection();
      ((HttpsURLConnection) urlConnection).setSSLSocketFactory(sslSocketFactory);

      urlConnection.setRequestProperty(samlBearerHeader.get(0).getName(),
samlBearerHeader.get(0).getValue());
      urlConnection.setRequestProperty(samlBearerHeader.get(1).getName(),
samlBearerHeader.get(1).getValue());

      urlConnection.connect();
      response.getWriter().println("Received from CF:");

      BufferedReader in = new BufferedReader(new InputStreamReader(
      urlConnection.getInputStream()));
      String inputLine;
      while ((inputLine = in.readLine()) != null)
                response.getWriter().append(inputLine);
            in.close();
    } catch (Exception e) {
        // Connectivity operation failed
```

```
        String errorMessage = e.getMessage();
        LOGGER.error("Connectivity operation failed", e);
        response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
errorMessage);
    }
  }
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    doGet(request, response);
  }
}
```

Requirements to the Application in the Cloud Foundry Environment

In the Cloud Foundry environment, you need an application following the XSA security model (protected with SAML, UAA service binding using JWT token needed, roles configured in the *xs-security.json*).

See:

- Application Router
- Building Roles and Role Collections for Applications

> **i Note**
>
> You can use the XSA Security Sample Application in GitHub (instructions and code) to develop and deploy an application compliant with the above requirements.

7.6.2.1 Create a Destination

Prerequisites

Before you create the required destination, you need to note down a few properties that will be used as values in the destination settings.

1. In the cloud cockpit, navigate to the subaccount in the Cloud Foundry environment.
2. Navigate to the application router.
3. Enter the *Environment Variables* section.
4. Note down somewhere the values of the following properties:
   - clientid
   - clientsecret
   - url

Context

Connect the first subaccount to the second subaccount by describing the source connection properties in a destination. For more information see Modeling Destinations [page 1072].

Procedure

1. Choose the global account and navigate to *Connectivity* > *Destinations*.
2. Choose *New Destination*.
3. In the new destination, provide the following information:

Field	Description
Name	Technical name of the destination. It can be used later on to get an instance of that destination. It must be unique for the global account. **i Note** For the purposes of the example listed in this document, use *pptest* as value.
URL	The URL of the protected resource in the Cloud Foundry environment. See Configuring Application URLs [page 1644]. Example: `https://<tenant-specific-route-for-your-business-app>.cfapps.eu10.hana.ondemand.com/`
Authentication	OAuth2SAMLBearerAssertion
Proxy Type	Internet

Audience	Copy the value of `entityID` property of the SAML 2.0 metadata representing your subaccount in the Cloud Foundry environment.
	
	→ Tip
	You can open the metadata of the subaccount in the Cloud Foundry environment using the following URL:
	`https://<your subaccount's subdomain>.authentication.<region host>/saml/metadata`
	Example:
	`https://demo.authentication.eu10.hana.ondemand.com/saml/metadata`
	For the `<region host>`, see Regions and API Endpoints Available for the Cloud Foundry Environment.
	
	Example of audience/entityID:
	`demo.aws-live-eu10`

Client Key	In the cloud cockpit, navigate to the application in the Cloud Foundry environment (*<path to your subaccount>* > *Spaces* > *<your space>* > *Applications* > *<your application>*). Open *Environment Variables*. Copy the value of the *clientid* property in *VCAP_SERVICES* > *xsuaa* > *credentials*.

Field	Description

Token Service URL	Get the **token service URL** from the SAML 2.0 metadata representing your subaccount in the Cloud Foundry environment. The **token service URL** is defined in the `Location` attribute of the element marked as `AssertionConsumerService`, like this :
	
	`<md:AssertionConsumerService Location="<Token Service URL>" Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI" index="1"/>`
	
	→ Tip
	
	You can open the metadata of the subaccount in the Cloud Foundry envirnoment using the following URL:
	
	`https://<your subaccount's subdomain>.authentication.<region host>/saml/metadata`
	
	Example:
	
	`https://demo.authentication.eu10.hana.ondemand.com/saml/metadata`
	
	For the `<region host>`, see Regions and API Endpoints Available for the Cloud Foundry Environment.
	
	Example of token service URL:
	
	`https://demo.authentication.eu10.hana.ondemand.com/oauth/token/alias/demo.aws-live-eu10`

Token Service User	In the cloud cockpit, navigate to the application in the Cloud Foundry environment (*<path to your subaccount>* > *Spaces* > *<your space>* > *Applications* > *<your application>*). Open *Environment Variables*. Copy the value of the *clientid* property in *VCAP_SERVICES* > *xsuaa* > *credentials* .
Token Service Password	In the cloud cockpit, navigate to the application in the Cloud Foundry environment (*<path to your subaccount>* > *Spaces* > *<your space>* > *Applications* > *<your application>*). Open *Environment Variables*. Copy the value of the *clientsecret* property in *VCAP_SERVICES* > *xsuaa* > *credentials* .
System User	Empty.

4. Save the changes.

7.6.2.2 Create Trust Between the Subaccounts

Procedure

1. In the cloud cockpit, log on with the Administrator user.
2. Save locally the service provider metadata of the subaccount in the Neo environment.

a. In SAP BTP cockpit, navigate to the subaccount in the Neo environment. See Navigate in the Cockpit
b. Navigate to *Security* > *Trust* > *Local Service Provider*.
c. Choose *Get Metadata* and save the metadata file representing this subaccount.

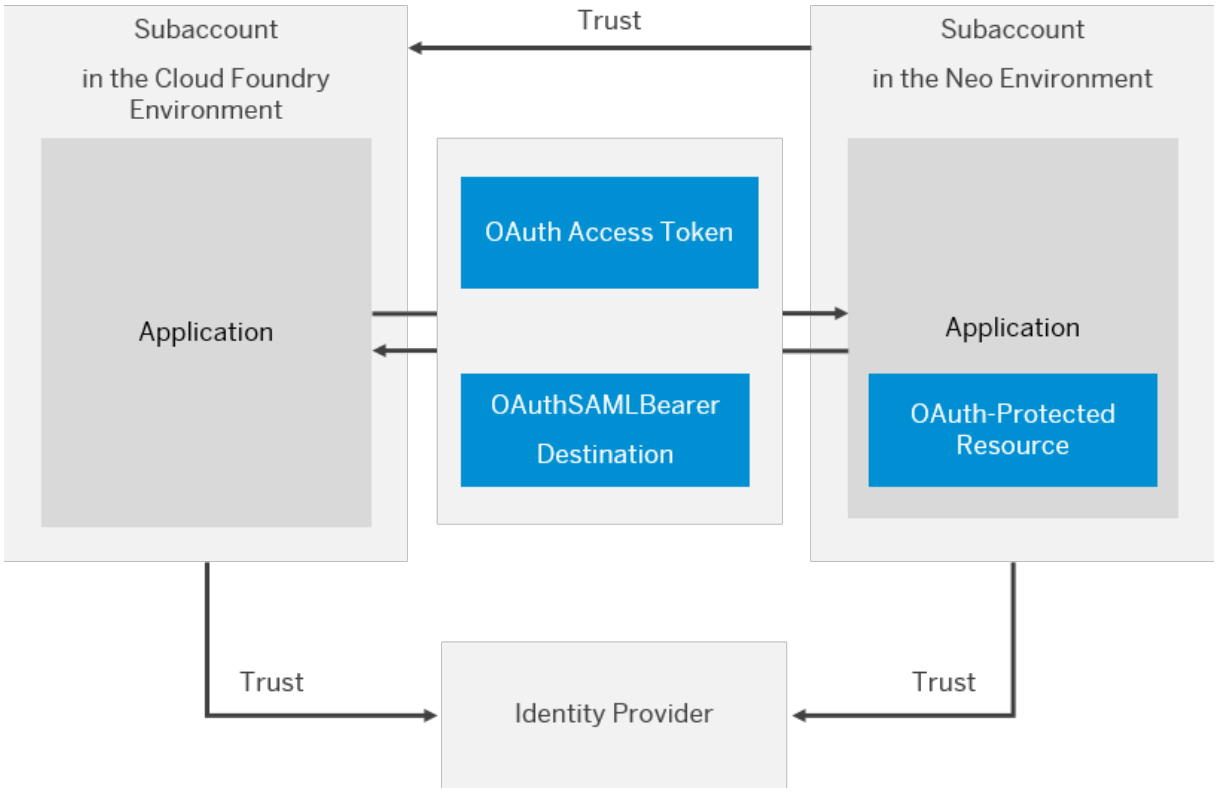
3. In the subaccount in the Cloud Foundry environment, create trust to the subaccount in the Neo environment.
   a. Navigate to the subaccount in the Cloud Foundry environment.
   b. Navigate to *Security* > *Trust Configuration*
   c. Choose *New Trust Configuration*.
   d. In the *Metadata* field, choose the *Upload* button.
   e. Upload the service provider metadata file representing the subaccount in the Neo environment.
   f. Save the new trust configuration.

   After this procedure, you can use the security context from the application in the Neo environment to the application in the Cloud Foundry environment. The assigned groups from the Neo environment can be used as role collections in the Cloud Foundry environment.

7.6.3 Principal Propagation from the Cloud Foundry to the Neo Environment

Enable an application in your subaccount in the Cloud Foundry environment to access an OAuth-protected application in a subaccount in the Neo environment without user login (and user interaction) in the second application. For this scenario to work, the two subaccounts need to be in mutual trust, and in trust with the same identity provider. The first application will propagate its logged-in user to the second application using an OAuth2SAMLBearer destination.

The graphic below illustrates the overall setup of the scenario.



Prerequisites

- You have a user account with *Administrator* role in both SAP BTP subaccounts. See Managing Member Authorizations in the Neo Environment [page 1301].
- You have a *custom* local service provider configuration (this means in *cloud cockpit* > *Security* > *Trust* > *Local Service Provider* > you have chosen *Configuration Type* > *Custom*) in your subaccount in the Neo environment. See Configure the Local Service Provider [page 1721].
- Both accounts have a trust configuration to the same identity provider. See:
  - Configure Trust to the SAML Identity Provider [page 1724] (for the Neo environment)
  - Establish Trust with Any SAML 2.0 Identity Provider in a Subaccount (for the Cloud Foundry environment)
- The application in the Neo environment is protected using OAuth 2.0. See OAuth 2.0 Service [page 1751].
- The application in the Cloud Foundry environment is bound to an instance of the following services:
  - *Destination Service*. See Create and Bind a Destination Service Instance.
  - *xsuaa*
- You have deployed both applications, each in the corresponding subaccount.

> **i Note**
>
> All configuration steps described in this tutorial are done using the cloud cockpit.

Contents

- Create Trust Between the Subaccounts [page 1829]
- Create an OAuth Client [page 1830]
- Create a Destination [page 1832]

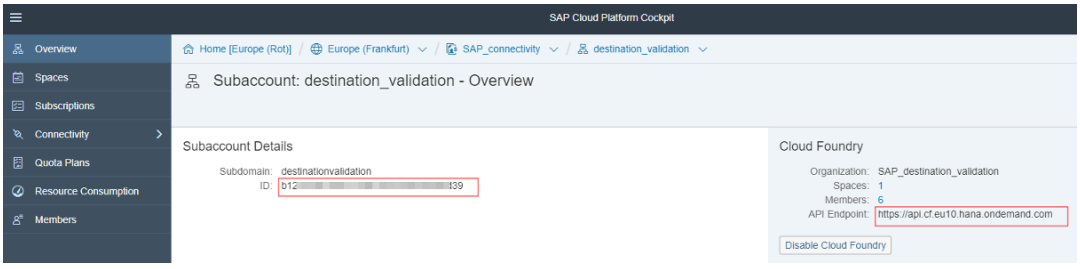
7.6.3.1 Create Trust Between the Subaccounts

Exchange keys and certificates between the subaccounts, and configure trust between them. This will enable the subaccounts to communicate using HTTP destinations.

Procedure

1. In the cloud cockpit, log on with the Administrator user.
2. Save locally the identifying X509 certificate of the subaccount in the Cloud Foundry environment.
   a. In SAP BTP cockpit, navigate to the subaccount in the Cloud Foundry environment.
   b. Navigate to *Connectivity* > *Destinations*.
   c. Choose *Download Trust* and save the X509 certificate identifying this subaccount.
3. In the subaccount in the Neo environment, create trust to the subaccount in the Cloud Foundry environment.
   a. Navigate to the subaccount in the Neo environment.
   b. Navigate to *Security* > *Trust* > *Application Identity Provider*.
   c. Choose *Add Trusted Identity Provider* and configure the new trust configuration settings:
      - In the *Name* field, enter the following:
        `<your Cloud Foundry API endpoint host>/<Cloud Foundry subaccount ID>`

        > → Tip
        >
        > You can view the **API endpoint host** and **subaccount ID** from *cloud cockpit* > *<your global account>* > *<your subaccount>* > *<your space>* > *Overview*.

      - In the *Signing Certificate* field, enter the X509 certificate of the Cloud Foundry account.

> **Note**
>
> Make sure you remove the `BEGIN CERTIFICATE` and `END CERTIFICATE` parts.

- Mark the *Only for IDP-initiated SSO* option.

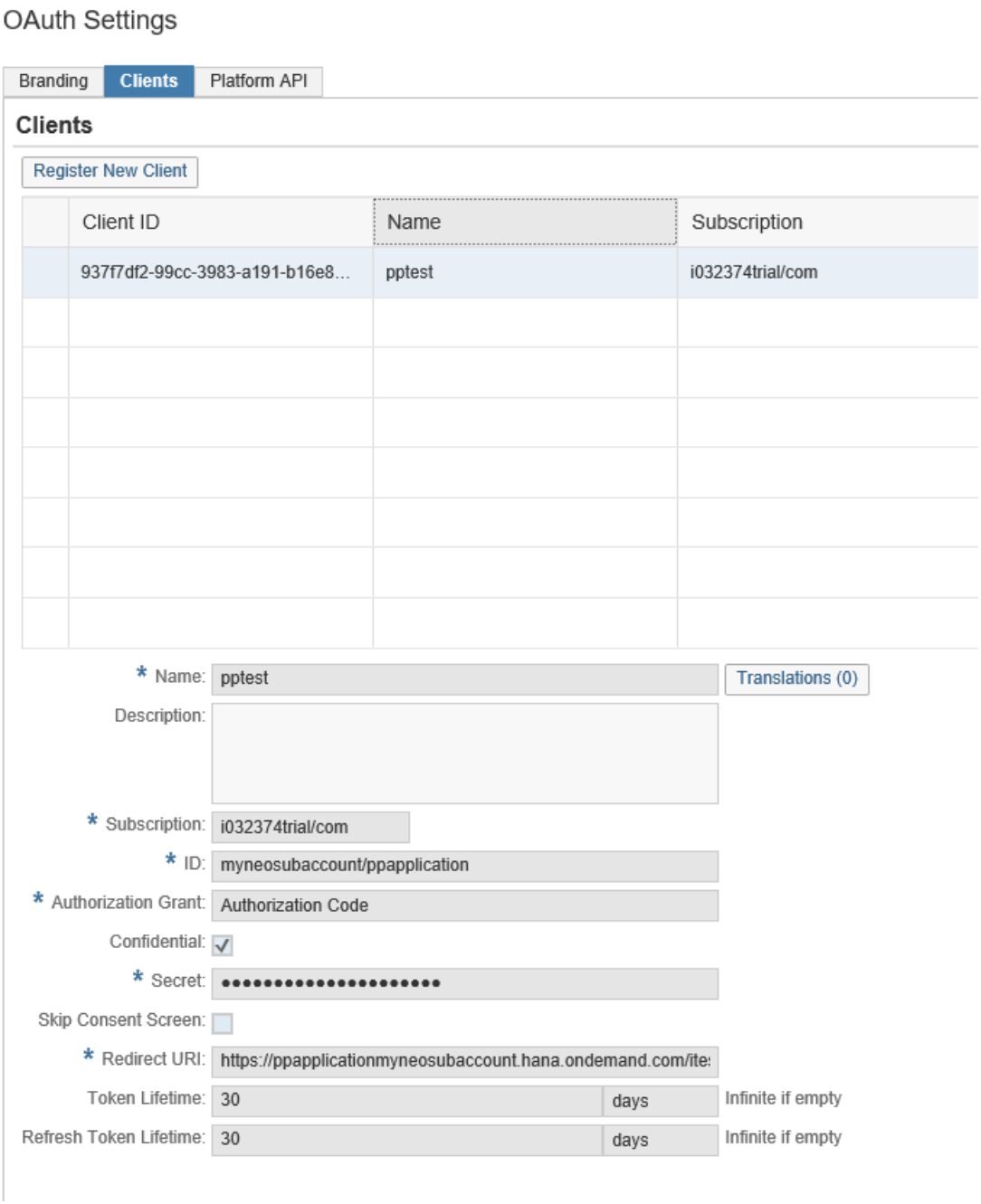
d. Save the new trust configuration.

7.6.3.2 Create an OAuth Client

You need an OAuth client to get an access token for the OAuth-protected resources in the application in the Neo environment.

Procedure

1. In the cloud cockpit, navigate to the subaccount in the Neo environment.
2. Navigate to *Security* > *OAuth* > *Clients*, and choose *Register New Client*.
3. Enter the following information:
   - *Name* - the OAuth client name. You will need to provide this name as value of the *Token Service User* property of the destination below.
   - *Authorization Grant* - choose the *Authorization Code* option.
   - Mark the *Confidential* option, and provide a secret (password).
4. Save the client.



i Note

When creating the required OAuthSAMLBearer destination later, you will need the following information from the OAuth client you created:

- *ID*
- *Secret*

7.6.3.3 Create a Destination

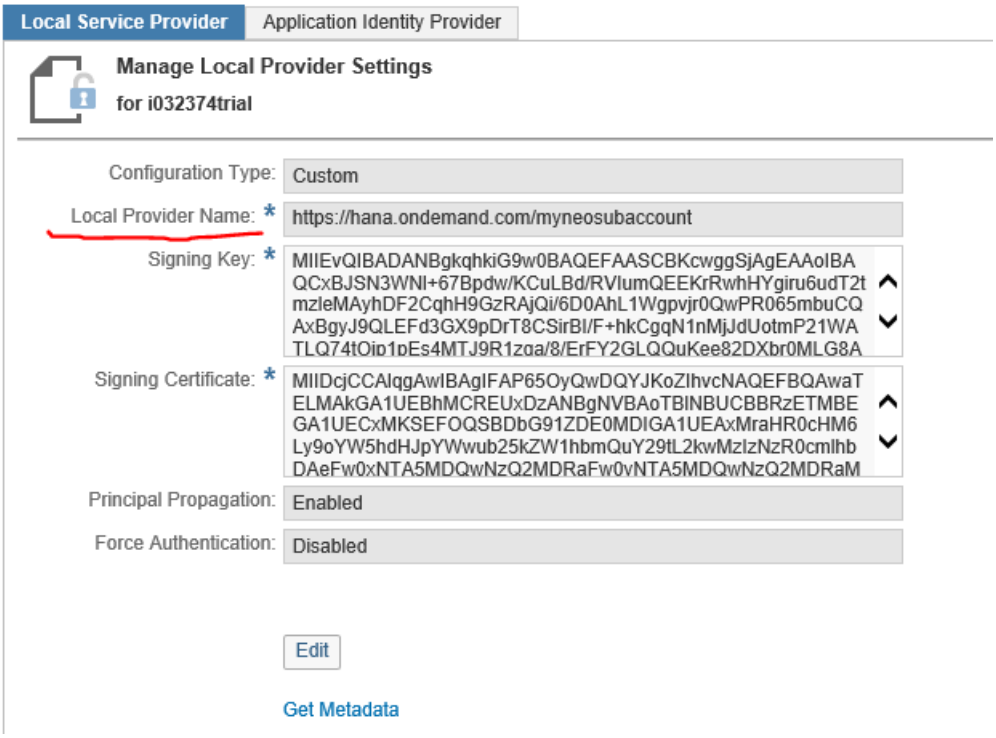
Context

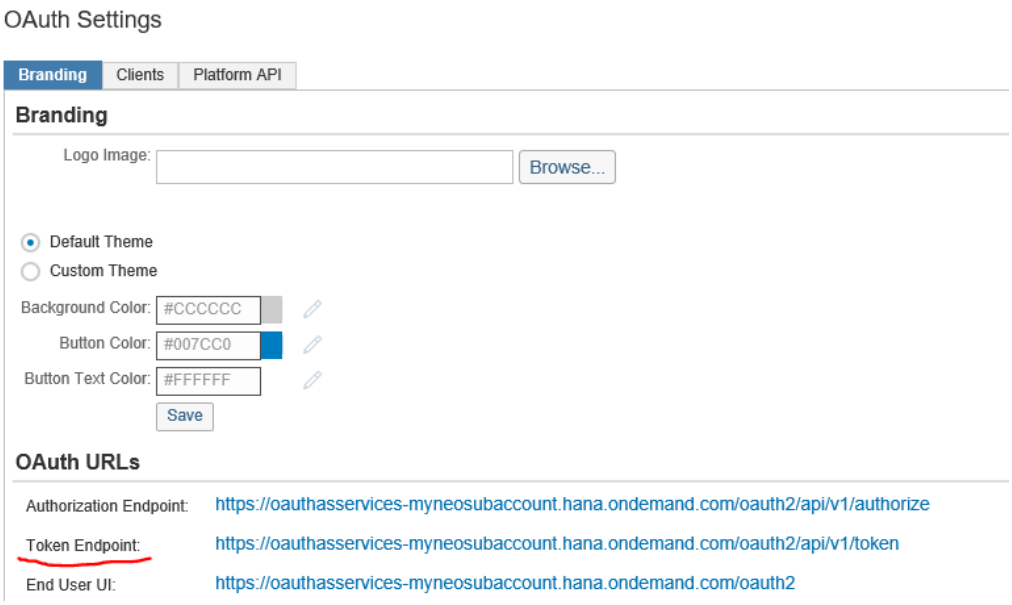
Connect the two subaccounts by describing the connection properties in a destination. For more information, see Modeling Destinations [page 1072].

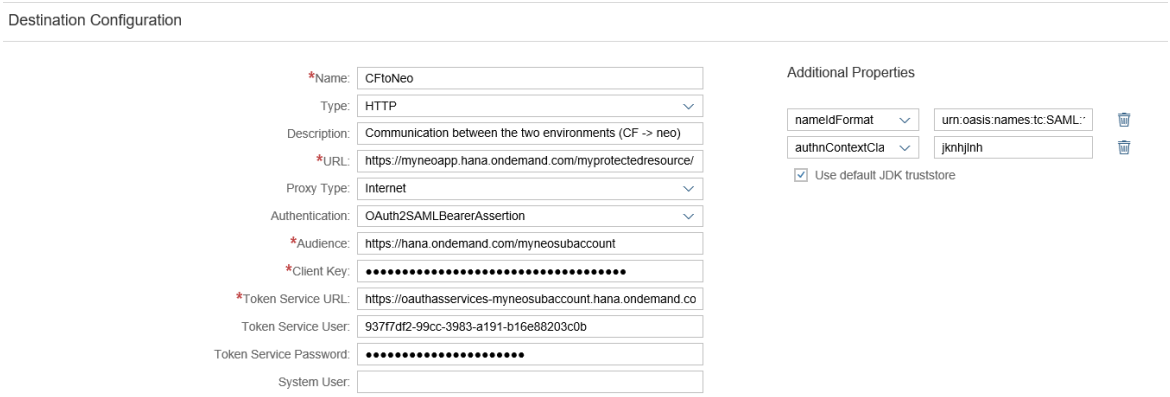
Procedure

1. Choose the subaccount in the Cloud Foundry environment, and navigate to *Connectivity* > *Destinations*.
2. Choose *New Destination*.
3. In the new destination, provide the following information:

Field	Description
Name	Technical name of the destination. It can be used later on to get an instance of that destination. It must be unique for the global account.
Description	Free-text description.
Type	HTTP
URL	The URL of the protected resource in the Neo environment. Example: `https://myneoapp.hana.ondemand.com/myprotectedresource/`
Authentication	OAuth2SAMLBearerAssertion
Proxy Type	Internet

Field	Description
Audience	The value of the local service provider name in the subaccount in the Neo environment. Copy the value from *cloud cockpit* > *<your Neo subaccount>* > *Security* > *Trust* > *Local Service Provider* > *Local Service Provider Name* >. 
Client Key	The ID of the OAuth client for the application in the Neo environment.

Field	Description
Token Service URL	Copy the value of *Token Endpoint* from the following place: cloud cockpit > *cloud cockpit* > *<your Neo subaccount>* > *Security* > *OAuth* > *Branding* >. 
Token Service User	The ID of the OAuth client for the application in the Neo environment.
Token Service Password	The secret from the OAuth client.
System User	Empty.
authnContextClassRef	*urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession*
nameIdFormat	*urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified* if the user ID will be propagated to the Neo application or *nameIdFormat = urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress* if the user email will be propagated to the Neo application.



4. Save the changes.

7.7 Protection from Web Attacks

To protect your applications from different kind of web attacks, Neo environment provides mechanisms for you to use with your applications.

Cross-Site Scripting (XSS)

XSS attacks allow the attacker to inject malicious code into a web application. See Protection from Cross-Site Scripting (XSS) [page 1836].

Cross-Site Request Forgery (CSRF)

With a CSRF attack, a malicious user tricks the victim's browser into executing an HTTP request on behalf of the valid user. See Protection from Cross-Site Request Forgery [page 1840].

Slow HTTP Attacks

Slow HTTP attacks happen when the attacker sends very slowly content in the server request one at a time to a Web server. If an HTTP request is not complete, or if the transfer rate is very low, the server keeps its resources busy waiting for the rest of the data. When the server's concurrent connection pool reaches its maximum, this creates a denial-of-service (DoS). Slow HTTP attacks are easy to execute because they require only minimal resources from the attacker.

> → Tip
>
> We recommend that you configure your applications to support connection timeout suitable for your network setup and application use case. Although you cannot completely avoid the threat of slow HTTP

attacks, using this configuration can lower the chance of successfully executed attacks. Make sure you don't make connection timeout too small so you allow enough timeout for handling normal network latency.

Configuring connection timeout is done at deploy time (using the `--connection-timeout` command parameter). See deploy [page 1422].

Parent topic: Security, Neo Environment [page 1671]

Related Information

Authorization and Trust Management in the Neo Environment [page 1672]
Platform Identity Provider [page 1746]
OAuth 2.0 Service [page 1751]
Keystore Service [page 1779]
Audit Logging in the Neo Environment [page 1810]
Principal Propagation [page 1816]
Data Protection and Privacy [page 1848]

7.7.1 Protection from Cross-Site Scripting (XSS)

This document describes how to protect SAP BTP applications from XSS attacks.

What is Cross-Site Scripting (XSS)

Cross-site Scripting (XSS) is the name of a class of security vulnerabilities that can occur in Web applications. It summarizes all vulnerabilities that allow an attacker to inject HTML Markup and/or JavaScript into the affected Web application's front-end.

XSS can occur whenever the application dynamically creates its HTML/JavaScript/CSS content, which is passed to the user's Web browser, and attacker-controlled values are used in this process. In case these values are included into the generated HTML/JavaScript/CSS without proper validation and encoding, the attacker is able to include arbitrary HTML/JavaScript/CSS into the application's frontend, which in turn is rendered by the victim's Web browser and, thus, interpreted in the victim's current authentication context.

How To Protect Applications from Cross-Site Scripting

There are several possibilities you can use to protect your application:

- Protecting applications using SAPUI5
- Protecting applications using the XSS Output Encoding Library

Protecting Applications Using SAPUI5

For SAPUI5 applications, XSS vulnerabilities can exist at different levels:

- Within the HTML page or custom data transports sent to the browser by the server
- Within the JavaScript Code of the application processing server responses
- Within the HTML renderers of SAPUI5 controls

For more information about the security measures implemented by SAPUI5, see Securing SAPUI5 Applications.

Protecting Applications from Domain Relaxation Risk

Domain Relaxation can occur in some on-premise UI technologies such as WebGUI, WebDynPro, and BSP UI. It uses the Same-Origin Policy, which allows a web browser to permit scripts contained in one web page to access data from another web page as long as they have the same root domain. This policy also prevents a malicious script on one page from obtaining access to sensitive data on another web page.

However, in some cases, an attacker can call the exposed service and use its domain relaxation feature, so that it shares its own application's root domain with your web pages. If this happens, the attacker has full access to all of your application resources.

To prevent this from happening, use the `--disable-application-url` parameter when creating a custom domain to block attackers from using your default application URL and prevent them from accessing your sensitive data. For more information, see add-custom-domain [page 1361].

Protecting Applications Using the XSS Output Encoding Library

> **i Note**
>
> Using the XSS output encoding library is given as an option that you can use for your applications. You can successfully use your custom or third-party XSS protection libraries that you have available.

SAP BTP provides an output encoding library that helps protecting from XSS vulnerabilities. It is a central library that implements several encoding methods for the different contexts.

In the application node, first retrieve the `com.sap.security.core.server.csi.IXSSEncoder` interface using `com.sap.security.core.server.csi.XSSEncoder.getInstance()`.

The interface provides methods for retrieving parameters or attributes, and for encoding and decoding data.

It also has various methods for different data types that should be encoded:

Data Type	Code Sample for Encoding
HTML / XML:	out = XSSEncoder.encodeHTML(in); / XSSEncoder.encodeXML(val);
JavaScript:	out = XSSEncoder.encodeJavaScript(val);
URL:	out = XSSEncoder.encodeURL(val);
CSS:	out = XSSEncoder.encodeCSS(val);

Installing the XSS Output Encoding Library

To use XSS output encoding API, you need to add it as library to the Dynamic Web Project. This is done with the following steps:

1. In the *Project Explorer* view, select the *persistence-with-jpa/WebContent/WEB-INF/lib* node.
2. From the context menu, choose *Import* > *General* > *File System* and choose *Next*.
3. Browse to your local directory where you downloaded and unpacked the SAP BTP SDK, select the *repository/plugins* directory (*/impl* directory if you are using the Tomcat 7 runtime), and choose *OK*.
4. Select the archive *com.sap.security.core.server.csi_1.x.y.jar* and choose *Finish*.

Using the XSS Output Encoding Library

In the following example, we demonstrate the use of the XSS Output Encoding API. The example has one HTML form that retrieves user input, which can contain malicious code:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
<title>Login Page</title>
<link rel="stylesheet" href="resources/login.css" />
<meta http-equiv="cache-control" content="no-cache"/>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    </head>
    <body>
      <div class="fields">
<form method="post" action="checkedInput.jsp">
          <span class="header">Enter your names:</span><br/>
          <table border="0">
            <tr>
              <td>First name: </td>
              <td><input name="firstname"/></td>
            </tr>
<tr>
              <td>Last name: </td>
              <td><input name="lastname"/></td>
</tr>
```

```
<tr>
<td></td>
<td><input type="submit" value="Submit"/></td>
            </tr>
          </table>
        </form>
      </div>
</body>
</html>
```

This form sends parameters to a second JSP:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import="com.sap.security.core.server.csi.IXSSEncoder" %>
<%@ page import="com.sap.security.core.server.csi.XSSEncoder" %>
<%@ page import="java.io.UnsupportedEncodingException" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Welcome</title>
<link rel="stylesheet" href="resources/login.css" />
<meta http-equiv="cache-control" content="no-cache"/> <meta http-equiv="Content-
Type" content="text/html; charset=UTF-8"/>
  </head>
<body>
<div class="fields">
      <form>
<%
String encodedName = validateInput(request.getParameter("firstname"));
out.println("<br>Hello, " + encodedName);
        String lastName = request.getParameter("lastname");
out.println("<br> Hacked: " + lastName);
%>
      </form>
    </div>
    <%!
      private String validateInput(String firstName) {
String encodedInput = null;
        IXSSEncoder xssEncoder = XSSEncoder.getInstance();
try {
encodedInput = xssEncoder.encodeHTML(firstName).toString();
        } catch (UnsupportedEncodingException e) {
          e.printStackTrace();
        } return encodedInput;
      }
    %>
  </body>
</html>
```

Even though the attacker might attempt to inject malicious code in both parameters - *firstname* and *lastname*, the firstname is protected, since it uses the output encoding library to neutralize all special symbols. However, the attack attempt will be successful for the lastname parameter since it is printed directly to the output. This is unsafe behavior and should be avoided.

7.7.2 Protection from Cross-Site Request Forgery

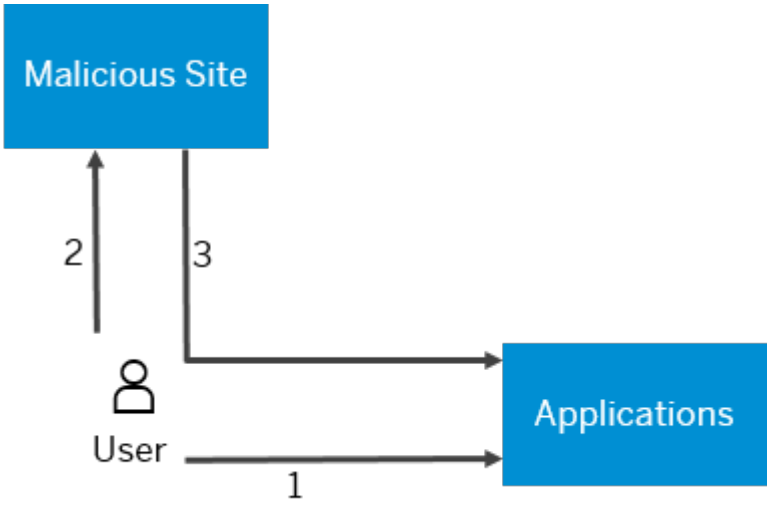
What is Cross-Site Request Forgery (CSRF)

Cross-site request forgery (CSRF or XSRF) is also known as one-click attack or session riding. The key step of the attack is that a malicious user tricks the victim's browser into executing an HTTP request on behalf of the valid user. As a result, a security sensitive action is performed on the server side. If the victim has already logged in the attacked site, the browser has valid session cookies and sends them automatically with subsequent requests. The server trusts these requests based on the valid cookies sent by the browser and confirms that the action has been initiated by the victim.

The predictability of the HTTP request is a prerequisite for the attacker to be able to insert a request in advance in order to make the browser execute it. Therefore, the common prevention to this attack is to embed a secret unpredictable token into the request, unique for each session or request.

The diagram that follows illustrates the CSRF process:

1. The victim logs in and creates session for the attacked web application.
2. The victim visits a malicious site in another browser window.
3. The malicious site makes request to the attacked application using the victim's session cookies.



How to Protect from CSRF

SAP BTP provides two CSRF protection approaches:

CSRF Protection Mechanism	Description	When to Use	How to Use
URL encoding approach	Based on the CSRF Prevention Filter provided by Apache Tomcat 7. The prevention mechanism is based on a token (a nonce value) generated on each request and stored in the session. The token is used to encode all URLs on the entry point sites. Upon request to a protected URL, the existence and value of the token is checked. The request is allowed to proceed only if the nonce from the token equals the one stored in the session. The prevention mechanism is applied for all URLs mapped to the filter except for specially defined entry points.	This is the most common CSRF protection. Use it for protecting resources that are supposed to be accessed via some sort of navigation. For example, if there is a reference to them in an entry point page (included in links/ post forms, and so on).	See Using the Apache Tomcat CSRF Prevention Filter [page 1842].
Custom header approach	Based on a secret token (a nonce value) generated on server side and stored in the session, but unlike the first approach, here the token is transported as a custom header of the HTTP requests.	Use it when URL encoding is not suitable. For example, when protecting resources that are requested only as REST APIs (one time requests that should be served independently from previous requests and are not included in links and HTML forms). The same approach is implemented in other SAP web application servers like AS ABAP and HANA XS, and is supported by SAP UI5. Common scenarios that can benefit from this approach are those using ODATA services, REST, AJAX, etc.	See Using Custom Header Protection [page 1844].
Custom CSRF filtering implementation	If you cannot use URL encoding or custom header protection, you can implement your custom CSRF filtering	Use it when implementing single logout (SLO) for SAP BTP applications. Due to redirects to the SAML 2.0 identity provider, you cannot use the out-of-the-box approaches listed here (custom header protection or URL encoding.	Logout [page 1697]

> **Note**
>
> These approaches cannot be applied together to protect one and the same web resource.

7.7.2.1 Using the Apache Tomcat CSRF Prevention Filter

Prerequisites

You have created a working Web application and have enforced authentication for it. See Authentication [page 1678].

For the purposes of this tutorial, an example application consisting of the following URLs will be used:

- `/home` - displays home page, and has links to `/doActionA` and `/doActionB`
- `/doActionA` - executes a security sensitive action A, and also has a link to `/doActionB`
- `/doActionB` - executes a security sensitive action B

Adding CSRF Prevention to a Web Application

1. Choose entry point URLs

Entry points are URLs used as a starting point for the navigation across the application. They are not protected against CSRF as requests to them will not be tested for the presence of a valid nonce. Entry points should meet the following criteria:

- Entry points are protected resources.
  - Entry points should not trigger any security sensitive actions.
  - Using the full set of entry points as navigation starting points, it should be possible to reach any link in the application that is being protected against CSRF.

Considering the example application, `/doActionA` and `/doActionB` are not plausible for entry points since they are state changing URLs. They should be protected against CSRF. Following the rules above, you could easily conclude that `/home` is best suited to be the entry point.

2. Define the filter in the application's web.xml

The CSRF Prevention Filter should be defined in the `web.xml` configuration file. Important init parameters are `entryPoints` and `nonceCacheSize`. The first parameter's value is a comma separated list of the entry points, identified in the previous step. In this case `/home`.

The second parameter, `nonceCacheSize`, should be used in case of parallel requests that might cause a new nonce to be generated, before the validation of an encoded URL. The `nonceCacheSize` parameters defines the number of previous values stored. The default number is 5.

The definition below will protect all URLs except for the entry point `/home`.

```
<filter>
  <filter-name>CsrfFilter</filter-name>
```

```
    <filter-class>org.apache.catalina.filters.CsrfPreventionFilter</filter-class>
    <init-param>
      <param-name>entryPoints</param-name>
      <param-value>/home</param-value>
    </init-param>
  </filter>
```

3. Define the filter mapping in the web.xm

The general recommendation is to enable the filter for all URLs using the pattern /*:

```
  <filter-mapping>
    <filter-name>CsrfFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

4. Encode all URL links

In the example application the URLs that should be encoded are `/protected/doActionA` and `/protected/doActionB` in `/protected/home`, and the `/protected/doActionB` URL in `/protected/doActionA`. To encode the URLs use `HttpServletResponse#encodeRedirectURL(String)` or `HttpServletResponse#encodeURL(String)`.

Here is the source for the `home.jsp`, mapped to `/protected/home`.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Home</title>
</head>
<body>
    This is a home page and I am an entry point.
    <br/>
    <%
        String urlActionA = "doActionA";
        String urlActionAEncoded = response.encodeURL(urlActionA);
    %>
    <form action="<%=urlActionAEncoded %>" method="POST">
        <input type="text" name="arg" value="A"/>
        <input type="submit" value="Do Action A"/>
    </form>
    <br/>
    <%
        // Encode URL for action B
        String urlActionB = "doActionB";
        String urlActionBEncoded = response.encodeURL(urlActionB);
    %>
    <form action="<%=urlActionBEncoded %>" method="POST">
        <input type="text" name="arg" value="B"/>
        <input type="submit" value="Do Action B"/>
    </form>
    <br/>
</body>
</html>
```

5. Adding new URLs to a CSRF protected application

In case a new URL needs to be added to the application later, for example, `/newlink`, then you should evaluate its need of CSRF protection. For example, if it executes a state changing action, it certainly should be protected. Depending on the case there are two possibilities:

- No CSRF protection is needed
  The URL should be defined as entry point. This is done by editing the `web.xml` and adding the new URL to the value of the init parameter `entryPoints` of the `CsrfPreventionFilter` from step 2. The new value should be separated with a comma.
- CSRF protection is needed.
  The `/newlink` URL should be encoded in all pages that use it as described in step 4.

All CSRF protected links that are used in the new page should be encoded, as described in step 4.

7.7.2.2 Using Custom Header Protection

Context

Custom header protection is one of the possible approaches for CSRF protection. It is based on adding a servlet filter that inspects state modifying requests for the presence of valid CSRF token. The CSRF token is transferred as a custom header and is valid during the user session. This kind of protection specifically addresses the protection of REST APIs, which are normally not accessed from entry point pages. Note that the CSRF protection is performed only for modifying HTTP requests (different from GET|HEAD or OPTIONS).

In a nutshell, the REST CSRF protection mechanism consists of the following communication steps:

1. The REST CLIENT obtains a valid CSRF token with an initial non-modifying "Fetch" request to the application.
2. The SERVER responds with the valid CSRF token mapped to the current user session.
3. The REST CLIENT includes the valid CSRF token in the subsequent modifying REST requests in the frame of the same user session.
4. The SERVER rejects all modifying requests to protected resources that do not contain the valid CSRF token.

Custom header CSRF protection mechanism requires adoption both in the client (JavaScript) and server (REST) parts of the Web applications.

To better illustrate the mechanism we'll use an example web application exposing the following REST APIs. We'll use the same example application throughout the document.

Number	Exposed with HTTP methods	REST API	Description	Type
1	GET	/services/list	Prints customers list in the output.	non-modifying
2	POST	/services/customers/removeCustomer	Removes the first item from the customers list.	modifying
3	POST	/services/customers/addCustomer	Adds a customer to the customers list.	modifying

1. In the REST Service

Prerequisites

You have created a working Web application and have enforced authentication for it, as described in Authentication [page 1678]. All CSRF protected resources should be protected with an authentication mechanism.

Procedure

In the application's `web.xml`, protect all REST APIs using the out-of-the-box CSRF filter available with the SAP BTP SDK.

> **i Note**
>
> You must have at least one non-modifying REST operation listed.

Identify all web application resources that have to be CSRF protected and map them to `org.apache.catalina.filters.RestCsrfPreventionFilter` (this class represents the out-of-the-box CSRF filter available with the SAP BTP SDK, so you do not need to instantiate/implement it) in the `web.xml`.

> **i Note**
>
> If you are using an older version of the SAP BTP rutime for Java, use the `com.sap.core.js.csrf.RestCsrfPreventionFilter` class instead. It delivers the same implementation as the other one. Namely, use that class with the following runtime versions:
>
> - Java Web 1.x lower than 1.98.22
> - Java EE Web Profile lower than 2.80.14
> - Java Web Tomcat 7 lower than 2.45.16

As a result, all modifying HTTP requests matching the given `url-pattern` would be CSRF validated, i.e. checked for the presence of the valid CSRF token.

Applications should expose at least one non-modifying REST operation to enable CSRF token fetch mechanism. In order to obtain the valid CSRF token, the clients need to make an initial fetch requests. That is why the non-modifying REST API is necessary. Requirements for the non-modifying REST API:

- Any GET/HEAD/OPTIONS requests to the URL shall not cause state modification.
- The URL should be mapped to the `RestCsrfPreventionFilter`
- The URL should be protected with authentication mechanism.

Example

The following example illustrates mapping a set of modifying REST APIs and one non-modifying REST API to the CSRF protection filter in the application's `web.xml` deployment descriptor:

```
<filter>
   <filter-name>RestCSRF</filter-name>
   <filter-class>org.apache.catalina.filters.RestCsrfPreventionFilter</filter-
class>
 </filter>
 <filter-mapping>
   <filter-name>RestCSRF</filter-name>
    <!— modifying REST APIs-->
    <url-pattern>/services/customers/removeCustomer</url-pattern>
    <url-pattern>/services/customers/addCustomer</url-pattern>
    <url-pattern>/services/customers/initCustomers</url-pattern>
    <!— non-modifying REST API-->
    <url-pattern>/services/customers/list</url-pattern>
 </filter-mapping>
```

2. In REST Clients

Procedure

1. Make a fetch request.

   As a first step, the REST client should obtain the valid CSRF token for the current session. For this it makes a non-modifying request and includes a custom header `"X-CSRF-Token: Fetch"`. The returned `[sessionid – csrf token]` pair should be cached and used in subsequent REST requests by the client. Another option is to send Fetch request before every REST request and thus to use the `[sessionid – csrf token]` pair only once.

   Example HTTP Request-Response flow:

   ```
   Client Request:
   GET /restDemo/services/customers/list HTTP/1.1
   X-CSRF-Token: Fetch
   Authorization: Basic dG9tY2F0OnRvbWNhdA==
   Host: localhost:8080

   Server Response:
   HTTP/1.1 200 OK
   Set-Cookie: JSESSIONID=4BA3D75B73B8C4591F1D915BA9C2B660; Path=/restDemo/;
   HttpOnly
   X-CSRF-Token: 5A44B387B75E54417F6C64FF3D485141
   ..
   ```

2. Use the cached `[sessionid – csrf token]` pair for subsequent REST requests.

   Subsequent modifying REST requests to the same application should include the valid jsessionid cookie and the valid `X-CSRF-Token` header.

   Example HTTP Request -Response flow:

   ```
   Client Request:
   ```

```
POST /restDemo/services/customers/removeCustomer HTTP/1.1
Cookie: JSESSIONID=4BA3D75B73B8C4591F1D915BA9C2B660
X-CSRF-Token: 5A44B387B75E54417F6C64FF3D485141
Authorization: Basic dG9tY2F0OnRvbWNhdA==
Host: localhost:8080

Server Response:
HTTP/1.1 200 OK
..
```

3. Handling error server responses

   The client should be prepared for the following server response:

   ```
   403 Forbidden
   X-CSRF-Token: Required
   ```

   It may occur in one of these cases:
   - Invalid or missing CSRF token in the request.
   - Expired session - after session expiration the `[sessionid – csrf token]` pair is no longer valid and it should be reinitialized by the client.
   - There are cases when the sessionid is changed by the server and the client should take into account such changes.

Exceptional Cases

Context

In small number of use cases the client is not able to insert custom headers in its calls to a REST API. For example file uploads via POST HTML FORM consuming a REST API. Only for such use-cases there is an additional capability to configure REST APIs for which the valid CSRF token will be accepted as request parameter (not only header). If there is a `X-CSRF-Token` header, it will be taken with preference over any parameter with the same name in the request.

> → Tip
>
> For security reasons we strongly recommend the following:
>
> - Use this approach only when the header approach cannot be applied.
> - Use only hidden post parameter with name `X-CSRF-Token`, and not query parameters.

Example configuration in the `web.xml` deployment descriptor:

```
<filter>
    <filter-name>CSRF</filter-name>
    <filter-class>org.apache.catalina.filters.RestCsrfPreventionFilter</filter-
class>
    <init-param>
        <param-name>pathsAcceptingParams</param-name>
        <param-value>/services/customers/acceptedPath1.jsp,/services/customers/
acceptedPath2.jsp
</param-value>
    </init-param>
</filter>
<filter-mapping>
```

```
    <filter-name>CSRF</filter-name>
    <url-pattern>/services/customers/*</url-pattern>
</filter-mapping>
```

7.8 Data Protection and Privacy

Data protection is associated with numerous legal requirements and privacy concerns. In addition to compliance with general data protection and privacy acts, it is necessary to consider compliance with industry-specific legislation in different countries.

SAP provides specific features and functions to support compliance with regard to relevant legal requirements, including data protection. SAP does not give any advice on whether these features and functions are the best method to support company, industry, regional, or country-specific requirements. Furthermore, this information should not be taken as advice or a recommendation regarding additional features that would be required in specific IT environments. Decisions related to data protection must be made on a case-by-case basis, taking into consideration the given system landscape and the applicable legal requirements.

> **i** Note
>
> SAP does not provide legal advice in any form. SAP software supports data protection compliance by providing security features and specific data protection-relevant functions. In many cases, compliance with applicable data protection and privacy laws will not be covered by a product feature. Definitions and other terms used in this document are not taken from a particular legal source.

> **⚠** Caution
>
> The extent to which data protection is supported by technical means depends on secure system operation. Network security, security note implementation, adequate logging of system changes, and appropriate usage of the system are the basic technical requirements for compliance with data privacy legislation and other legislation.

Generic Fields

You also need to make sure that no personal data enters the system in an uncontrolled or non-purpose related way, for example, in free-text fields, or customer extensions.

SAP BTP

This documentation covers personal data relating to SAP BTP accounts and data stored in databases by SAP BTP. SAP BTP offers a number of capabilities, that is, services, buildpacks, application, and so on. Here we cover the core platform. For more information about data protection and privacy for capabilities you have purchased, see the data protection and privacy documentation for those capabilities.

To view the services consumed by a global account:

1. Navigate to the global account to which you'd like to view members.
   For more information, see Navigate to Orgs and Spaces.
2. In the navigation area, choose *Entitlements*.

This documentation is written with the data protection officer of a company in mind. The processes described here may be required for a data protection officer or an administrator of the user accounts for your tenants or even business users of the tenants. In particular the processes for business users are described here so that you in your role of data protection officer or account administrator can communicate them to your business users if required.

- Global account users are stored in platform identity provider or a tenant of SAP Cloud Identity Services - Identity Authentication.
- Platform users are stored in platform identity provider, a tenant of SAP Cloud Identity Services - Identity Authentication, or your own identity provider.
- Business users are stored in a tenant of SAP Cloud Identity Services - Identity Authentication or your own identity provider.

Glossary for Data Protection and Privacy [page 1850]
: The following terms are general to SAP products. Not all terms may be relevant for SAP BTP.

Change Logging and Read-Access Logging [page 1851]
: Change logging records changes to personal data, while read-access logging records access to sensitive personal data. You may be required to gather this information for auditing purposes or legal requirements.

Personal Data Record [page 1852]
: A personal data record is a collection of data relating to a data subject. A data privacy specialist may be required to provide such a record or an application may offer a self-service.

Deletion [page 1852]
: The processing of personal data is subject to applicable laws related to the deletion of this data when the specified, explicit, and legitimate purpose for processing this personal data has expired. If there is no longer a legitimate purpose that requires the retention and use of personal data, it must be deleted.

Consent [page 1858]
: SAP BTP supports you in collecting and managing the consent of data subjects in the following ways:

Parent topic: Security, Neo Environment [page 1671]

Related Information

Authorization and Trust Management in the Neo Environment [page 1672]
Platform Identity Provider [page 1746]
OAuth 2.0 Service [page 1751]
Keystore Service [page 1779]
Audit Logging in the Neo Environment [page 1810]
Principal Propagation [page 1816]
Protection from Web Attacks [page 1835]

7.8.1 Glossary for Data Protection and Privacy

The following terms are general to SAP products. Not all terms may be relevant for SAP BTP.

Term	Definition
Blocking	A method of restricting access to data for which the primary business purpose has ended.
Business purpose	The legal, contractual, or in other form justified reason for the processing of personal data to complete an end-to-end business process. The personal data used to complete the process is predefined in a purpose, which is defined by the data controller. The process must be defined before the personal data required to fulfill the purpose can be determined.
Consent	The action of the data subject confirming that the usage of his or her personal data shall be allowed for a given purpose. A consent functionality allows the storage of a consent record in relation to a specific purpose and shows if a data subject has granted, withdrawn, or denied consent.
Data subject	Any information relating to an identified or identifiable natural person ("data subject"). An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier, or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural, or social identity of that natural person.
Deletion	Deletion of personal data so that the data is no longer available.
End of business	Defines the end of active business and the start of residence time and retention period.
End of purpose (EoP)	The point in time when the processing of a set of personal data is no longer required for the primary business purpose, for example, when a contract is fulfilled. After the EoP has been reached, the data is blocked and can only be accessed by users with special authorizations (for example, tax auditors).
End of purpose (EoP) check	A method of identifying the point in time for a data set when the processing of personal data is no longer required for the primary business purpose. After the EoP has been reached, the data is blocked and can only be accessed by users with special authorization, for example, tax auditors.
Personal data	Any information relating to an identified or identifiable natural person ("data subject"). An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier, or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural, or social identity of that natural person.
Purpose	The information that specifies the reason and the goal for the processing of a specific set of personal data. As a rule, the purpose references the relevant legal basis for the processing of personal data.

Term	Definition
Residence period	The period of time between the end of business and the end of purpose (EoP) for a data set during which the data remains in the database and can be used in case of subsequent processes related to the original purpose. At the end of the longest configured residence period, the data is blocked or deleted. The residence period is part of the overall retention period.
Retention period	The period of time between the end of the last business activity involving a specific object (for example, a business partner) and the deletion of the corresponding data, subject to applicable laws. The retention period is a combination of the residence period and the blocking period.
Sensitive personal data	A category of personal data that usually includes the following type of information: • Special categories of personal data such as data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership and the processing of genetic data, biometric data, data concerning health, sex life or sexual orientation or personal data concerning bank and credit accounts • Personal data subject to professional secrecy • Personal data relating to criminal or administrative offenses • Personal data concerning insurances and bank or credit card accounts
Technical and organizational measures (TOM)	Some basic requirements that support data protection and privacy are often referred to as technical and organizational measures (TOM). The following topics are related to data protection and privacy and require appropriate TOMs, for example: • Access control: Authentication features • Authorizations: Authorization concept • Read access logging • Transmission control/communication security • Input control/change logging • Availability control • Separation by purpose: Subject to the organizational model implemented and must be applied as part of the authorization concept.

7.8.2 Change Logging and Read-Access Logging

Change logging records changes to personal data, while read-access logging records access to sensitive personal data. You may be required to gather this information for auditing purposes or legal requirements.

Use the appropriate API to retrieve logs.

For the Neo environment, see Audit Log Retrieval API Usage for the Neo Environment [page 1810].

> **i Note**
>
> For any applications you develop, you must ensure they include logging functions. SAP BTP does not provide audit logging functions for custom developments.

Logs are deleted regularly.

7.8.3 Personal Data Record

A personal data record is a collection of data relating to a data subject. A data privacy specialist may be required to provide such a record or an application may offer a self-service.

To see the personal data that is used for membership management within SAP BTP, access the cloud cockpit.

1. Navigate to the global account to which you'd like to view members.
   For more information, see Navigate to Orgs and Spaces.
2. In the navigation area, choose *Members*.

To see the personal data that is used for application logging within SAP BTP, access the cloud cockpit.

For more information, see Using Logs in the Cockpit or Analyze Logs from the Cockpit in the Application Logging service documentation.

If you do not use your own identity provider for identity federation, you can view the profiles available in SAP Cloud Identity Services - Identity Authentication.

For more information, see Information Report in the SAP Cloud Identity Services - Identity Authentication documentation.

For SAP BTP Cloud Foundry environment, the User Account and Authentication service creates shadow users to issue tokens for their corresponding users.

For more information about viewing shadow users, see the User Management (SCIM) API on SAP API Business Hub.

For all other services, which persist data, such as databases or document services, retrieve the data you stored with the same APIs, protocols, or languages you used to store the data.

To view the services used in a global account, choose *Entitlements* in the navigation area.

7.8.4 Deletion

The processing of personal data is subject to applicable laws related to the deletion of this data when the specified, explicit, and legitimate purpose for processing this personal data has expired. If there is no longer a legitimate purpose that requires the retention and use of personal data, it must be deleted.

When deleting data in a data set, all referenced objects related to that data set must be deleted as well. Industry-specific legislation in different countries also needs to be taken into consideration in addition to general data protection laws. After the expiration of the longest retention period, the data must be deleted.

When accounts expire, we delete your data barring legal requirements that SAP retains your data. If your organization has separate retention requirements, you are responsible for saving this data before we terminate your account.

- For trial accounts in the Cloud Foundry environment, your account expires after 365 days.
- Productive accounts expire based on the terms of your contract.

To deactivate or delete users, see Erasure in the SAP Cloud Identity Services - Identity Authentication documentation.

For SAP BTP Cloud Foundry environment, the User Account and Authentication service creates shadow users to issue tokens for their corresponding users. Data privacy regulations or policies may require you to delete this data, for example, when the user has left your organization.

For more information about deleting shadow users, see:

- Delete Shadow Users for Data Protection and Privacy Using the Cockpit [page 1853]
- Delete Shadow Users for Data Protection and Privacy Using APIs [page 1856]

For all other services which persist data, you can retrieve the data you stored with the same APIs, protocols, or languages which you used to store the data.

To view the services used in a global account, choose *Entitlements* in the navigation area.

SAP BTP Data Retention Manager is a service available for the Cloud Foundry environment that helps you to identify data subjects for deletion as well as maintain rules for residence and retention.

For more information, see SAP BTP Data Retention Manager.

We maintain backups of the data for disaster recovery. When your account is deleted, we may have this data in our backup system for the length of our backup cycle.

> **i Note**
>
> If your data is stored outside SAP BTP, we cannot guarantee that your data does not get reintegrated if you are pushing such data to our systems. You are responsible for terminating such integrations.
>
> We cannot restore data you have in your local system.

Related Information

Global Account Termination

7.8.4.1 Delete Shadow Users for Data Protection and Privacy Using the Cockpit

Data privacy regulations or policies may require you to delete this data, for example, when the user has left your organization. SAP BTP cockpit offers an application to select and delete shadow users.

Prerequisites

You have the required authorizations.

For more information, see Security Administration: Managing Authentication and Authorization or Role Collections and Roles in Global Accounts, Directories, and Subaccounts [Feature Set B].

Context

> **i Note**
>
> When handling personal data, consider the legislation in the various countries and regions where your organization operates. After the data has passed the end of purpose, regulations might require you to delete the data. For more information on data protection and privacy, see the related link.

The User Account and Authentication service stores user-related data records in the form of shadow users. The UAA uses the information of the shadow users to issue tokens that refer to the specific user. If automatic shadow user creation is enabled, the UAA creates the shadow users when the user authenticates. Otherwise, the UAA creates the shadow user as soon as you assign the user a role collection. These conditions apply to platform users and business users. For more information about shadow users, see the Cloud Foundry documentation.

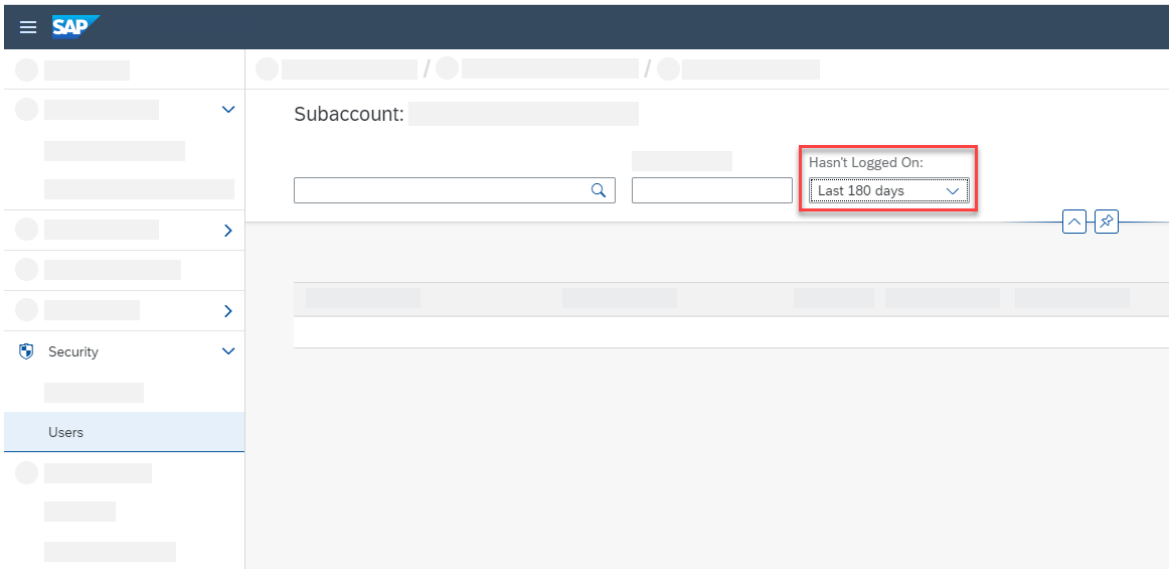
> **i Note**
>
> You can also delete users using the User Management (SCIM) API.
>
> For more information, see Delete Shadow Users for Data Protection and Privacy Using APIs [page 1856].

Procedure

1. Open the SAP BTP cockpit.
2. Go to your global account, directory, or subaccount (see Navigate in the Cockpit).
3. Choose *Security* > *Users*.

   > **i Note**
   >
   > For directories, choose *Users*. There is no *Security* menu item.

4. From the list of users, select the users to delete.



Filter for Users Who Haven't Logged On

→ Recommendation

Users, who have left your organization, haven't logged on recently. Use the *Hasn't Logged On* filter option to find users who haven't logged on in the last 180 days.

The last logon date you filter for varies from situation to situation. The filter value must meet the following requirements:

- Be long enough that the filter doesn't catch users who are simply on vacation or don't work regularly in the system.
- Not be so long that the filter no longer meets your data protection and privacy requirements.

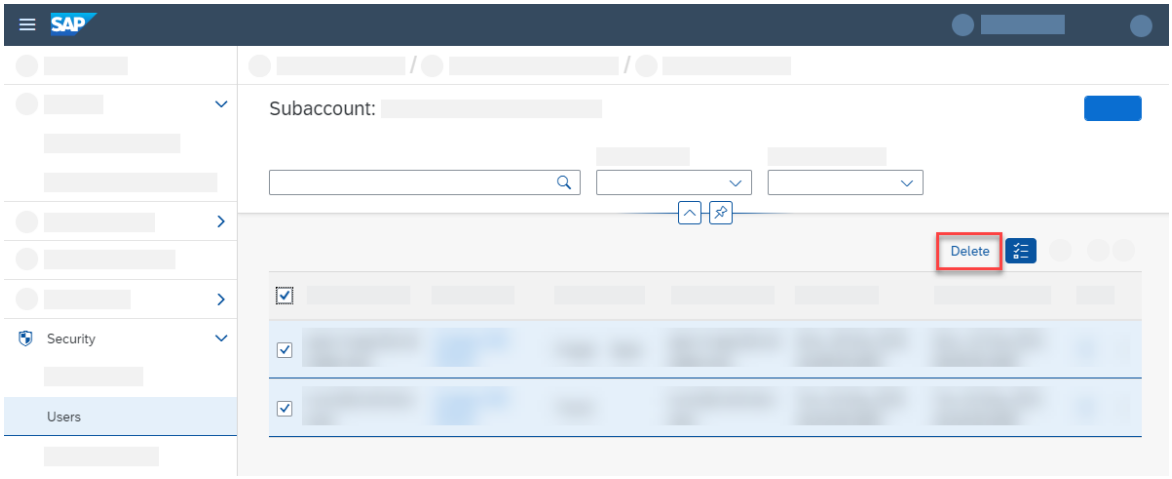
Think about users on parental leave or on a longer leave of absence, too.

⚠ Caution

You can't undo the deletion of a user. When you delete a user, you delete any direct assignments of role collections to that user. You also potentially invalidate any application referencing that user.

For example, an application saves some data relating to a user. When you delete the user, that data points to an ID that no longer exists. You can recreate the user, but the new user has a different ID, even though other attributes, such as e-mail, first name, and last name, are identical.

5. Delete the selected users.



Delete Selected Users

Related Information

Delete Users
Data Protection and Privacy [page 1848]
Switch Off Automatic Creation of Shadow Users
Shadow Users in the Cloud Foundry Documentation

7.8.4.2 Delete Shadow Users for Data Protection and Privacy Using APIs

Data privacy regulations or policies may require you to delete this data, for example, when the user has left your organization. To delete shadow users using APIs, set up access to the API and then use the SCIM REST APIs to retrieve and delete the users.

Prerequisites

The security administrator must have the following scopes:

- `xs_user.read`
- `xs_user.write`

Context

> **i Note**
>
> When handling personal data, consider the legislation in the various countries and regions where your organization operates. After the data has passed the end of purpose, regulations might require you to delete the data. For more information on data protection and privacy, see the related link.

The User Account and Authentication service stores user-related data records in the form of shadow users. The UAA uses the information of the shadow users to issue tokens that refer to the specific user. If automatic shadow user creation is enabled, the UAA creates the shadow users when the user authenticates. Otherwise, the UAA creates the shadow user as soon as you assign the user a role collection. These conditions apply to platform users and business users. For more information about shadow users, see the Cloud Foundry documentation.

> **i Note**
>
> Administrators can also delete users using the SAP BTP cockpit. For more information, see Delete Users.

Procedure

1. Enable API access to your subaccount.

   For more information, see Access UAA Admin APIs.
2. Use the *GET* method for the User Management (SCIM) API of the SAP Authorization and Trust Management service to get a list of users.

   For more information about the User Management (SCIM) API, see https://api.sap.com/package/authtrustmgmnt on *SAP API Business Hub*.

   The API returns the list of users in JSON format.
3. Use your own tools to sort and identify the users in the JSON response to delete.

   > **→ Recommendation**
   >
   > Users, who have left your organization, haven't logged on recently. Use the `lastLogonTime` parameter to find users who haven't logged on in the last 180 days. The `lastLogonTime` parameter is in UNIX time.
   >
   > The last logon time you filter for varies from situation to situation. The filter value must meet the following requirements:
   >
   > - Be long enough that the filter doesn't catch users who are simply on vacation or don't work regularly in the system.
   > - Not be so long that the filter no longer meets your data protection and privacy requirements.
   >
   > Account for users on parental leave or on a longer leave of absence, too.

> ⚠ Caution
>
> You can't undo the deletion of a user. When you delete a user, you delete any direct assignments of role collections to that user. You also potentially invalidate any application referencing that user.
>
> For example, an application saves some data relating to a user. When you delete the user, that data points to an ID that no longer exists. You can recreate the user, but the new user has a different ID, even though other attributes, such as e-mail, first name, and last name, are identical.

4. Use the *DELETE* method and the list of user IDs to delete the shadow users one at a time.

   For each user, call the `Users` endpoint with the delete method and include the user ID in the path. For example:

   ```
   curl --location --request DELETE 'https://
   api.authentication.eu20.hana.ondemand.com/Users/a0a67e6f-c4b5-41e5-
   b871-6fa181d46599' \
   --header 'Content-Type: application/x-www-form-urlencoded' \
   --header 'Accept: application/json' \
   --header 'Authorization: bearer eyJqdGkiOiJmZmZl…'
   ```

   For more information about the User Management (SCIM) API, see https://api.sap.com/package/authtrustmgmnt on *SAP API Business Hub*.

Related Information

Data Protection and Privacy [page 1848]
https://docs.cloudfoundry.org/uaa/uaa-concepts.html#%23shadow
Switch Off Automatic Creation of Shadow Users
Delete Shadow Users for Data Protection and Privacy Using the Cockpit [page 1853]

7.8.5 Consent

SAP BTP supports you in collecting and managing the consent of data subjects in the following ways:

SAP Cloud Identity Services - Identity Authentication provides tools to manage privacy policies and terms of use agreements.

For more information, see Configuring Privacy Policies and Configuring Terms of Use in the SAP Cloud Identity Services - Identity Authentication documentation.

See also Consent in the SAP Cloud Identity Services - Identity Authentication documentation.

8 Getting Support, Neo Environment

Use SAP Community, get guided answers, or explore SAP Support Portal.

Prerequisites

Before you report an incident, ensure the following:

Check Platform Status

Visit SAP Trust Center. You can check:

- availability by service on the *Cloud Status* tab page;
- availability by region on the *Data Center* tab page.

For more information about selected platform incidents, see Root Cause Analyses.

Check Tools Versions

Make sure that you've the latest versions of the tools you're using (recommended), or at least the versions you're using are still supported. For more information, see SAP Development Tools.

Procedure

To report an incident (issue) in SAP Support Portal, proceed as follows:

1. Log On to SAP ONE Support Launchpad

Context

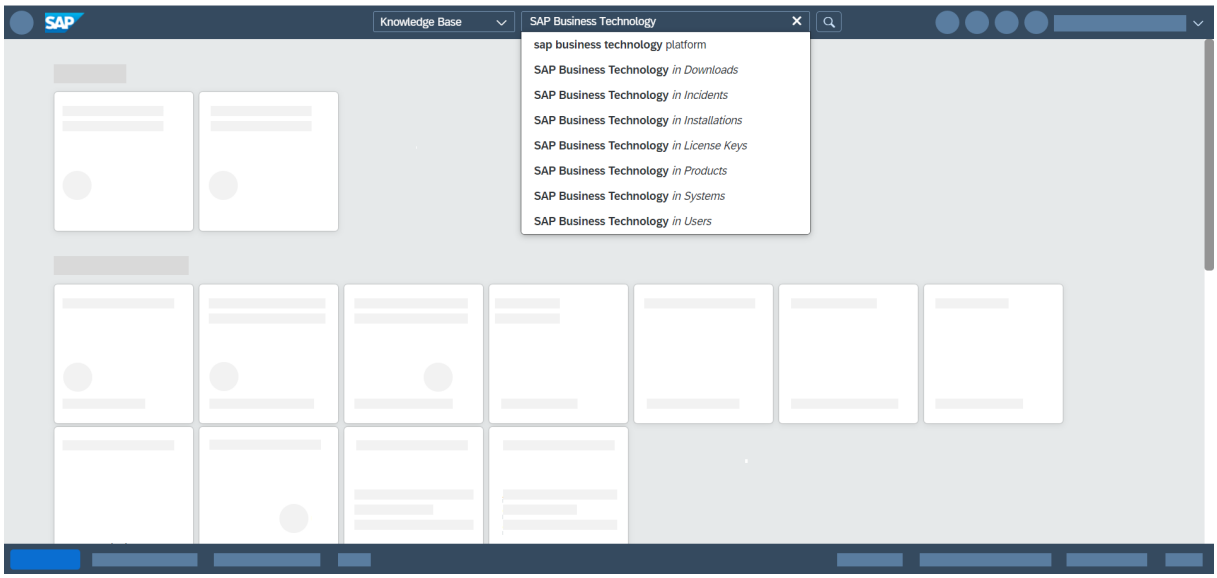
Open SAP ONE Support Launchpad.

2. (Optional) Search in the Knowledge Base for Existing Solutions

Context

You can search in Knowledge Base for existing solutions in:

- SAP Notes & KBAs
- SAP Community
- SAP Community Wiki
- SAP Support Portal
- Product Documentation
- Guided Answers



3. Choose Report an Incident

Context

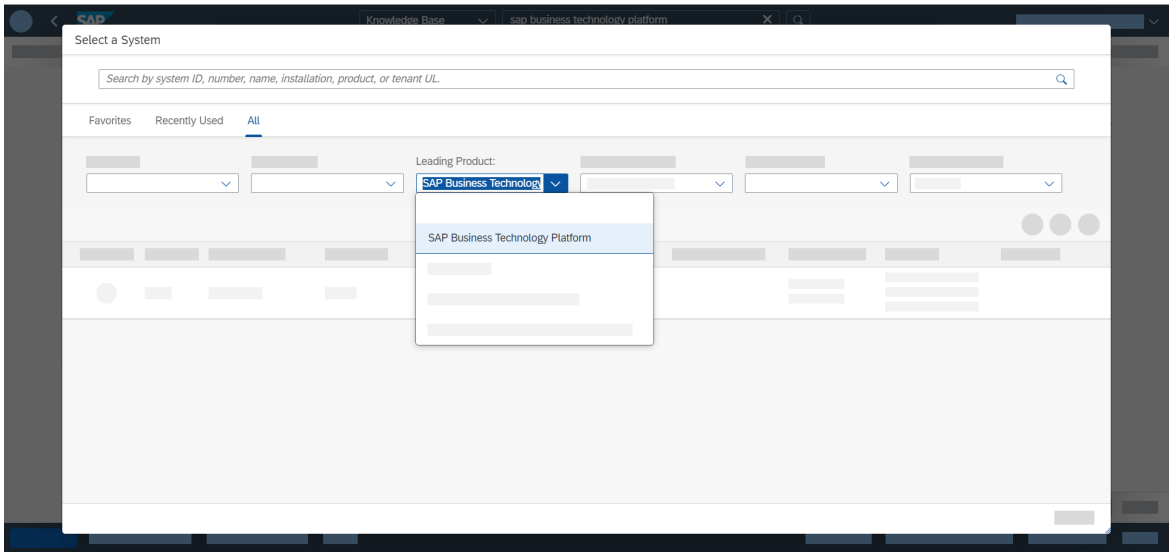
> ⚠ Caution
>
> If your S-user is assigned to several customer numbers, select a customer number from the drop-down list.

4. Provide System Information

Procedure

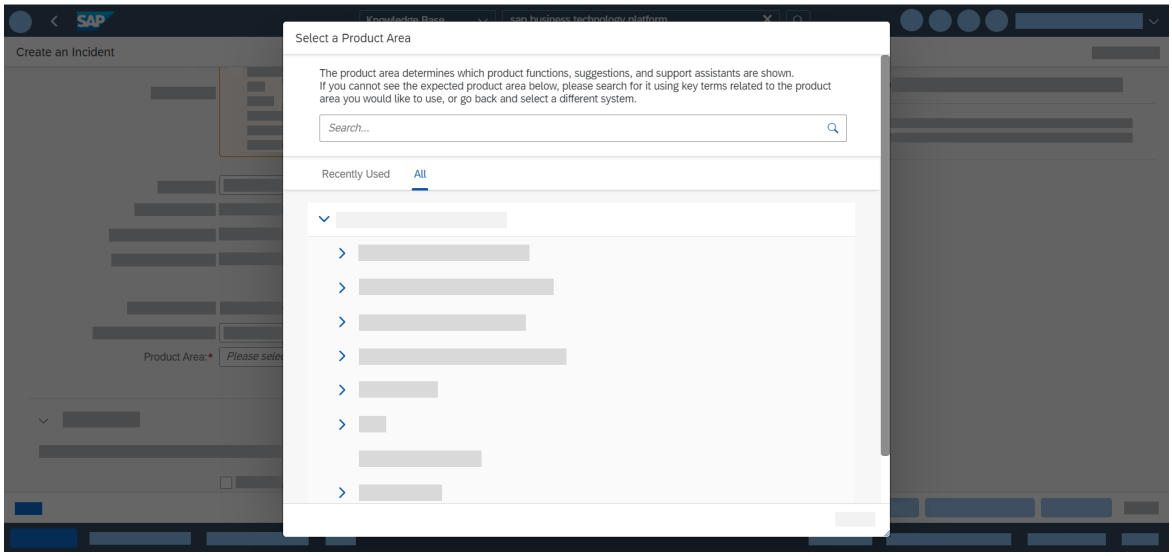
1. Select system.

   Specify the affected system. To report an issue for a service, filter your systems by the leading product: SAP Business Technology Platform or a service name.



2. Select product area.

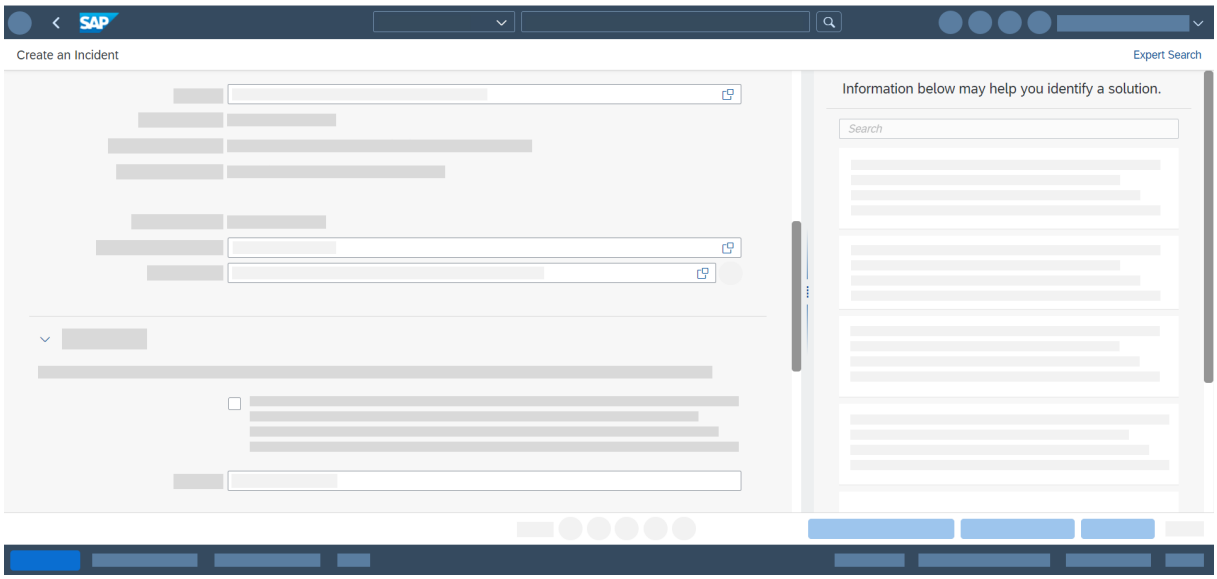
   You can select the area for your service or for a related product.



   For more information, see KBA 3023272.

Results

You can see recommended knowledge resources for the selected product area on the right-side panel.



> **i** Note
>
> When you specify the correct system, the correct support SLA is applied to your case.
>
> Not choosing the appropriate system and product area may negatively affect the processing of the incident.

5. Provide Description

Procedure

1. Enter a subject.
2. To help support staff process your issue as fast as possible, fill in the *Description* field:

   Enter the steps to reproduce the issue.

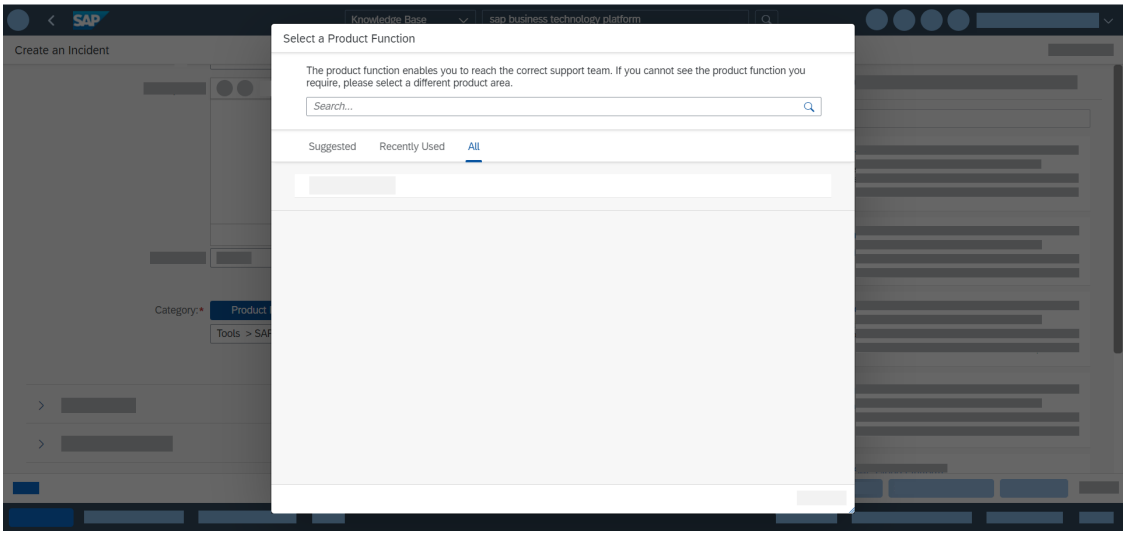
   Provide:

   - Region and global account name. In the cockpit, open the affected subaccount, and copy the URL.
   - Java application name and URL (if the problem is related to Java applications). In the cockpit, open the respective Java application's Overview page.
   - Database-related details based on your environment and infrastructure provider (if the problem is related to SAP HANA). See Providing Details for SAP HANA Service Database Problems.

   You can see SAP Knowledge Base Articles and SAP Notes recommended by Incident solution matching service as potential solutions. See Incident Solution Matching

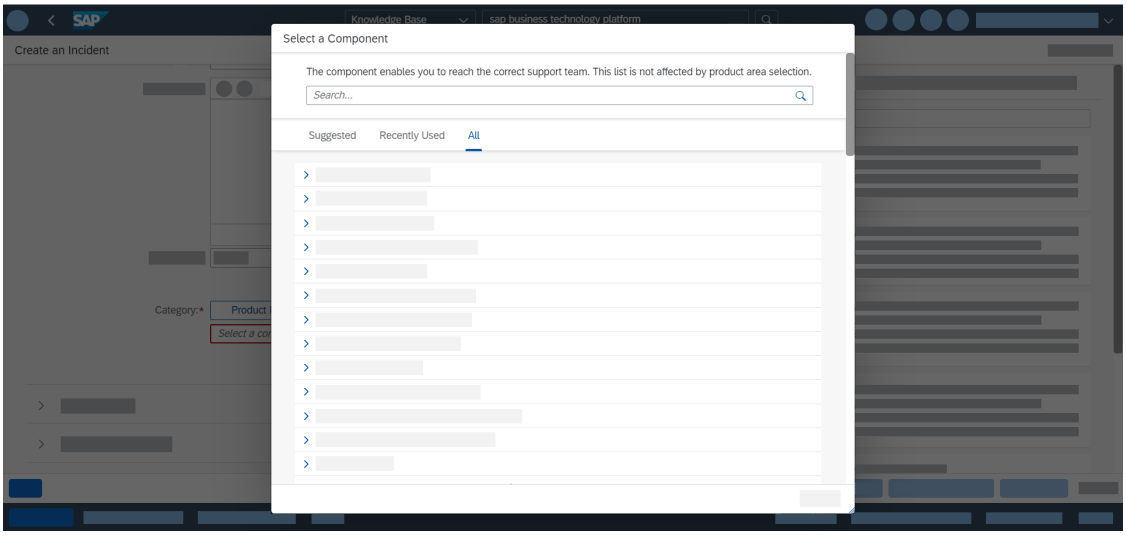
3. Select a category. You can either select:

   - *Product Function*
     Product functions are prefiltered by the selected system and product area. When you select the right product function, the component is automatically assigned to the incident.

     For more information, see KBA 3023272.

   - *Component*
     Select the component name of the area that best fits your issue. Selecting the right component directs the issue to the corresponding support team. To check the complete list of components, see Support Components (prefiltered for the Neo environment).

6. Provide Attachments

Context

Upload attachments complying with the required size and file types.

7. Set Priority

Procedure

- If you set a high or very high priority, you must also describe the business impact of the incident.
- (Optional) Define any additional contacts, apart from the reporter (who is filled in automatically).

8. Choose Support Channel

Context

To submit an incident, you can use one of the following support channels:

Procedure

- Review & Submit Incident

  The incident is submitted and the communication is carried out through the incident.
- Schedule an Expert

  You can book a 30-minute meeting slot with a support expert. An incident is automatically created and used to document the session. For more information, see KBA 2651981.
- Expert Chat

  You can start a chat with an expert. An incident is created to document your communication with the expert. For more information, see KBA 2570790.

Results

> **i** Note
>
> If you have problems creating and sending an incident, or your incident isn't processed as fast as you need, contact the 24/7 phone hotlines. See SAP Note 560499.

Related Information

Getting Support, Neo Environment [page 1859]

8.1 Providing Details for Database Problems in the Neo Environment

If your problem is related to a database, the details you need to provide differ depending on the environment or infrastructure provider the database is provisioned in.

Environment	Infrastructure Provider	Details You Need to Provide	How to Find the Details You Need
Neo	SAP regions	Region and global account name	In the cockpit, open the affected subaccount, and copy the URL.
		The database or schema ID	1. In the cockpit, navigate to the subaccount the database is provisioned in. 2. In the navigation area, go to *SAP HANA / SAP ASE* > *Databases & Schemas* and copy the ID of the database or schema from there.

Related Information

Providing Details for SAP HANA Service Database Problems

8.2 Gather Support Information

The Eclipse tools come with a wizard for gathering support information in case you need help with a feature or operation (during deploying/debugging applications, logging, configurations, and so on).

Context

The wizard collects the information in a ZIP file, which can be later sent to the support team. This way, the support developers can get better understanding of your environment and process the issue faster.

Procedure

1. From the Eclipse IDE, choose *Help* > *Collect Support Information*.
2. The launched wizard lists the default components to be collected, depending on the tools you've installed. If you need the support team to look at specific resources, expand the *Additional Data* section and select the relevant items.

   > **i Note**
   >
   > If you select *Screenshot*, your currently open Eclipse windows and views are snapped as a picture and added to the ZIP file. Make sure you don't reveal sensitive information.

3. In the *File Name* field, specify the ZIP file name and location.
4. Choose *Finish*.

Next Steps

You can create a support ticket, attach the files to it, and send it to the corresponding support team. For more information, see Getting Support, Neo Environment [page 1859].

8.3 Platform Updates and Notifications in the Neo Environment

SAP BTP is a dynamic product, which has continuous production releases (updates). To get notifications for the new features and fixes every release, subscribe at the SAP Community wiki by choosing the *Watch* icon.

Reasons for Region Updates

Regions are updated in the following cases:

- **Biweekly updates** (standard) - aligned with the contractual obligations to customers and partners. Such updates usually don't affect productive applications, because most services support zero downtime maintenance. See Service Level Agreement for SAP Cloud Services.
- **Immediate updates** - fixes required for bugs that affect productive application operations, or due to urgent security fixes. In some cases, this might lead to downtime or application restart, for which the application groups receive a notification.
- **Major upgrades** - happen rarely, in a bigger maintenance window, - up to four times per year. For the time frames of the services' major upgrades, see Service Level Agreement for SAP Cloud Services. We let you know about these upgrades four weeks in advance.

Announcements and Subscriptions

You can follow the availability of the platform at SAP Trust Center. You can check:

- the availability by service on *Cloud Status* tab page;
- the availability by region on the *Data Center* tab page.

To get notifications for updates and downtimes, subscribe at the Cloud System Notification Subscriptions application. Create a subscription by specifying Cloud Product, Cloud Service, and Notification Type. For more information, see Cloud System Notification Subscriptions User Guide.

Related Information

What's New

8.4 Operating Model in the Neo Environment

An operating model clearly defines the separation of tasks between SAP and the customer during all phases of an integration project.

Neo environment and its services have been developed on the assumption that specific processes and tasks are the responsibility of the customer. The following table contains all processes and tasks involved in operating the platform and the services and specifies how the responsibilities are divided between SAP and the customer for each individual task. It does not include the operation of systems and devices residing at operational facilities owned by the customer or any other third party, as these are the customer's responsibility.

Changes to the operating model defined for the services in scope are published using the *What's New* (release notes) section of the platform. Customers and other interested parties must review the product

documentation on a regular basis. If critical changes are made to the operating model, which require action on the customer side, an explicit notification is sent by e-mail to the affected customers.

It is not the intent of this document to supplement or modify the contractual agreement between SAP and the customer for the purchase of any of the services in scope. In the event of a conflict, the contractual agreement between SAP and the customer as set out in the Order Form, the General Terms and Conditions of SAP Cloud Services, the supplemental terms and conditions, and any resources referenced by those documents always takes precedence over this document.

The responsibilities for operating the Neo environment are listed in the service catalog below.

Service Catalog

Process	Task	SAP	Customer
Communication Management	Appoint an English-speaking contact person and communicate the name to SAP. This is required to ensure timely processing of configuration change requests affecting the customer system, interacting with SAP for efficient incident processing, and other interaction between SAP and the customer.		x
	Subscribe to the communication channels offered by SAP for receiving prompt information about any service disruptions, critical maintenance activities affecting the customer system, and change requests requiring action on the customer side.		x
	Inform the customer about service disruptions and critical maintenance activities affecting the customer system.	x	
Asset Management	Management of the hardware and infrastructure resources in the region, from acquisition through disposal. This includes the request and approval process, procurement management, life-cycle management, and disposal management.	x	

Process	Task	SAP	Customer
	Protect IT assets such as systems, network, and data from threats that arise from unauthorized physical access or physical influence on those assets.	x	
Provisioning	Provisioning of resources and systems to customers in accordance with the ordered package and subscriptions. This includes the allocation and provisioning of technical (physical and virtual) resources, such as storage, network, compute units, systems, and database hosts, the deployment of the SAP application software and the proper initial configuration of quotas, service subscriptions, permissions, and trust configuration.	x	
	Provide quota according to the ordered package and subscriptions that can be used to enable resources and services (for example, subscribing to a service).		x
Change Management	Apply regular product increments, as well as corrections to the infrastructure, systems, and services to avoid incidents with minimal possible disruption of normal operations. Ensure that all platform changes (such as updates of the Java runtime or operating system patches, but not of the customer applications) are evaluated, authorized, prioritized, planned, tested, implemented, documented, and reviewed prior to implementation.	x	

Process	Task	SAP	Customer
	Perform updates of the infrastructure, systems, and services in a bi-weekly cycle if required. Respectively, for selected services (such as SAP HANA and SAP ASE), offer self-services for applying controlled updates of new versions. Emergency changes, for example, triggered by Incident Management processes, have accelerated testing, approval, and implementation.	x	
	• Ensure prompt delivery of security patches via the Security Patch Management process. • Provision new database systems and Java applications with the latest patched versions. • Apply the security patches on live customer systems (Java application runtimes or databases), in case the patches do not require downtime, or if the vulnerable system puts at risk SAP or other customers. • Inform the customers about the availability of security patches.	x	
	Adopt the latest patches or updates via the available self-services and by restarting applications when necessary. For example, when a security issue arises.		x

Process	Task	SAP	Customer
Incident Management	Process incidents reported by the customer according to the Service Level Agreement. The incident is recorded and prioritized in the incident tracking system (BCP). Monitor the status and progress of the incident throughout its whole lifecycle and give regular status updates to the customer.	x	
	In the event of incidents, make reasonable effort to support end users and manage their incidents, to explore self-help tools to find already documented solutions, and to liaise with SAP support in the event of new problems to ensure timely processing of incidents affecting the resources in the customer account.		x
	Confirm incident resolution in the incident tracking system (BCP).		x
Service Requests	Process service requests reported by the customer according to the Service Level Agreement. The service request is recorded and prioritized in the service request tracking system (BCP). Monitor the status and progress of the service request throughout its whole lifecycle and give regular status updates to the customer.	x	
	Confirm service request completion in the service request tracking system (BCP).		x

Process	Task	SAP	Customer
Backup & Restore	Perform a backup of the database systems hosted in the subaccount. A database log backup is done every 10 minutes and stored on the primary storage. Every 2 hours the logs are transferred from primary to secondary storage. Full data backup is done every day.	x	
	Restore previously backed-up data to recover to a consistent state. Verify the completeness of the restored data based on log files created during the recovery and smoke tests to verify the system's consistency.	x	
	Give regular status updates to the customer throughout the entire restore procedure.	x	
	Collaborate with SAP to ensure timely processing of data restores if required.		x
	Validate logical integrity and consistency of the restored data.		x
User Access Management	Manage users, permissions, and security configurations within the subaccount.		x
System Monitoring	Ensure availability of the customer system according to the Service Level Agreements as agreed in the contractual agreement between SAP and the customer, by active monitoring, prompt issue detection, and incident prevention.	x	
	Monitor the resource consumption (memory, CPU, storage) to detect issues in technical operations.	x	

Process	Task	SAP	Customer
Malware Management	Ensure that the infrastructure and platform services are free of viruses, spam, spyware, and other malicious software. If malware is detected, an auto-notification is generated, which is assessed and resolved by the operator.	x	
Application Management	Design, develop, deploy, configure, maintain, and operate the application within the subaccount. This includes maintaining a staged environment for application delivery (if required), application resource management, and managing application availability and performance.		x
	Provide infrastructure, tools, and application programming interfaces for the lifecycle management and operations of the application in the subaccount.	x	
	Regularly adopt the latest versions of the tools for lifecycle management and operations offered at the SAP Development Tools site.		x
Network Management	Manage the network isolation of the subaccounts provisioned to the customer.	x	
	Operate the network infrastructure transparently for customers, ensuring elasticity, high availability, and security.	x	
	Create and manage own Web domain for the application in the subaccount to ensure data isolation.		x

Process	Task	SAP	Customer
Penetration Testing	Inform SAP about any penetration testing that shall be performed for the customer account and ask for their approval. Testing is not allowed on any resources shared with other customers. The results, if any, from the test are to be treated strictly as the confidential information of SAP and the customer are not to be shared with any person or entity without explicit written authorization from SAP. Customers are required to share the results with SAP and work together with SAP operations to mitigate or remedy any security issues.		x
Decommissioning	Ensure the secure deletion of data and/or hardware disposal. This includes the disassembling of systems along with peripherals and their removal from the region. Before dismantling and handover for further use or return to the vendor, the data is wiped securely from the system.	x	

8.5 Support Components

A list of support components for SAP BTP services and tools. Filter for the service you want to find the component for or have a look at the tools and software logistics section.

> **i** Note
>
> The table below lists the support components for services. If you are looking for components of tools or issues related to software logistics, see Additional Components [page 1968].

Service Components

Service Availability

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Agentry	Develop and run Agentry metadata-driven mobile applications.	MOB-CLD-AGS	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Brazil (São Paulo) Canada (Toronto) Japan (Tokyo) Russia (Moscow) UAE (Dubai) KSA (Riyadh)		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Alert Notification	Create and receive real-time alerts about your services	BC-CP-LCM-ANS	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Japan (Tokyo) Canada (Toronto) Australia (Sydney) KSA (Riyadh) UAE (Dubai) Brazil (São Paulo) Russia (Moscow)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Alert Notification	Create and receive real-time alerts about your services	BC-CP-LCM-ANS	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA) Japan (Tokyo) Singapore Australia (Sydney) Canada (Montreal) Brazil (São Paulo) Europe (Frankfurt) EU Access South Korea (Seoul)	Yes	Available
Alert Notification	Create and receive real-time alerts about your services	BC-CP-LCM-ANS	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo) Australia (Sydney)	Yes	Available
Alert Notification	Create and receive real-time alerts about your services	BC-CP-LCM-ANS	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
API Management	Expose your data and processes as APIs and manage their lifecycles.	OPU-API-OD	Neo	Integration Suite	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) Australia (Sydney) Japan (Tokyo) Brazil (São Paulo) Russia (Moscow) Canada (Toronto) UAE (Dubai) KSA (Riyadh)	Yes	Available
API Management	Expose your data and processes as APIs and manage their lifecycles.	OPU-API-OD	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) US East (VA) Singapore Japan (Tokyo) Australia (Sydney) Brazil (São Paulo) Canada (Montreal)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
API Management	Expose your data and processes as APIs and manage their lifecycles.	OPU-API-OD	Cloud Foundry	Integration Suite	Azure	Japan (Tokyo) Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo)	Yes	Available
API Management	Expose your data and processes as APIs and manage their lifecycles.	OPU-API-OD	Cloud Foundry	Integration Suite	Alibaba	China (Shanghai)**	Yes	Available
Mobile App and Device Management	Manage your mobile devices.	MOB-SEC	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* US East (Ashburn) US West (Chandler) Australia (Sydney) Japan (Tokyo) US East (Sterling)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Application Autoscaler	Automatically increase or decrease the number of application instances.	BC-CP-CF-AUTO-SCALE	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore Canada (Montreal)	Yes	Available
Application Autoscaler	Automatically increase or decrease the number of application instances.	BC-CP-CF-AUTO-SCALE	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo) Australia (Sydney)	Yes	Available
Application Autoscaler	Automatically increase or decrease the number of application instances.	BC-CP-CF-AUTO-SCALE	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available
Application Autoscaler	Automatically increase or decrease the number of application instances.	BC-CP-CF-AUTO-SCALE	Cloud Foundry	Extension Suite - Development Efficiency	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Application Logging Service	Create, store, access, and analyze application logs.	BC-NEO-LOG	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available
Application Logging Service	Create, store, access, and analyze application logs.	BC-CP-CF-APPLOG	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Application Logging Service	Create, store, access, and analyze application logs.	BC-CP-CF-APPLOG	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo) Australia (Sydney)	Yes	Available
Application Logging Service	Create, store, access, and analyze application logs.	BC-CP-CF-APPLOG	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available
Application Logging Service	Create, store, access, and analyze application logs.	BC-CP-CF-APPLOG	Cloud Foundry	Extension Suite - Development Efficiency	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Authorization and Trust Management Service	Manage application authorizations and trusted connections to identity providers.	BC-NEO-SEC-IAM	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Authorization and Trust Management Service	Manage application authorizations and trusted connections to identity providers.	BC-XS-SEC	Cloud Foundry	Extension Suite - Development Efficiency	AWS	US East (VA) Europe (Frankfurt) Europe (Frankfurt) EU Access Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal)	Yes	Available
Authorization and Trust Management Service	Manage application authorizations and trusted connections to identity providers.	BC-XS-SEC	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Australia (Sydney) Singapore Japan (Tokyo)	Yes	Available
Authorization and Trust Management Service	Manage application authorizations and trusted connections to identity providers.	BC-XS-SEC	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Authorization and Trust Management Service	Manage application authorizations and trusted connections to identity providers.	BC-XS-SEC	Cloud Foundry	Extension Suite - Development Efficiency	Alibaba	China (Shanghai)**	Yes	Available
Big Data Services	Ready-to-use, fully managed Hadoop and Spark.	BC-NEO-BDS		Extension Suite - Development Efficiency	SAP	This service does not run on standard SAP Cloud Platform regions. Check out this note for further details.		Available
Blockchain Application Enablement	Deliver blockchain-based services on any connected blockchain network.	BC-BCS-VAS	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) US East (VA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Build	Create interactive prototypes based on end-user feedback without code writing.	MOB-UIA-BLD-ADM	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) UAE (Dubai) Australia (Sydney) Japan (Tokyo) Russia (Moscow) Canada (Toronto) Brazil (São Paulo) KSA (Riyadh)	Yes	Available
Business Entity Recognition	Detect and highlight entities from unstructured text using machine learning.	CA-ML-BER	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Business Rules	Enrich cloud offerings with decision modeling, management, and execution service.	LOD-BPM-RUL	Neo	Extension Suite - Digital Process Automation	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) Russia (Moscow) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Canada (Toronto) Brazil (São Paulo) Australia (Sydney) Japan (Tokyo) UAE (Dubai) KSA (Riyadh)		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Business Rules	Enrich cloud offerings with decision modeling, management, and execution service.	LOD-BPM-RUL	Cloud Foundry	Extension Suite - Digital Process Automation	AWS	Europe (Frankfurt) Australia (Sydney) US East (VA) Singapore Japan (Tokyo) Brazil (São Paulo) South Korea (Seoul) Europe (Frankfurt) EU Access Canada (Montreal)	Yes	Available
Business Rules	Enrich cloud offerings with decision modeling, management, and execution service.	LOD-BPM-RUL	Cloud Foundry	Extension Suite - Digital Process Automation	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo) Australia (Sydney)	Yes	Available
Business Rules	Enrich cloud offerings with decision modeling, management, and execution service.	LOD-BPM-RUL	Cloud Foundry	Extension Suite - Digital Process Automation	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Process Integration	Seamlessly orchestrate business processes and integrate data in real time.	LOD-HCI-PI-OPS-PROV	Neo	Integration Suite	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Russia (Moscow) Canada (Toronto) Brazil (São Paulo) KSA (Riyadh) UAE (Dubai)		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Connectivity	Establish connections between cloud applications and on-premise systems.	BC-NEO-CON	Neo	Integration Suite	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Connectivity	Establish connections between cloud applications and on-premise systems.	BC-CP-CON-CF	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal)	Yes	Available
Connectivity	Establish connections between cloud applications and on-premise systems.	BC-CP-CON-CF	Cloud Foundry	Integration Suite	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo) Australia (Sydney)	Yes	Available
Connectivity	Establish connections between cloud applications and on-premise systems.	BC-CP-CON-CF	Cloud Foundry	Integration Suite	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Connectivity	Establish connections between cloud applications and on-premise systems.	BC-CP-CON-CF	Cloud Foundry	Integration Suite	Alibaba	China (Shanghai)**	Yes	Available
Continuous Integration and Delivery	Configure and run predefined pipelines for continuous integration and delivery.	BC-CP-CF-CICD	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA) Japan (Tokyo)	Yes	Available
Data Integration	Integrate data between on-premise and cloud applications on a scheduled (batch) basis.	LOD-HCI-DS	Neo	Integration Suite	SAP	UAE (Dubai) Australia (Sydney) Europe (Rot)* Japan (Tokyo) Russia (Moscow) KSA (Riyadh) US West (Colorado Springs)		Available
Credential Store	Store and retrieve credentials such as cryptographic keys and passwords.	BC-CP-CF-SEC-CPG	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Australia (Sydney) US East (VA) Brazil (São Paulo) Singapore Canada (Montreal) Japan (Tokyo)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Credential Store	Store and retrieve credentials such as cryptographic keys and passwords.	BC-CP-CF-SEC-CPG	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Japan (Tokyo) Europe (Netherlands) US West (WA) US East (VA) Singapore	Yes	Available
Credential Store	Store and retrieve credentials such as cryptographic keys and passwords.	BC-CP-CF-SEC-CPG	Cloud Foundry	Extension Suite - Development Efficiency	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Custom Domain Service	Configure and expose your application under your own domain.	BC-NEO-INFR	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Custom Domain Service	Configure and expose your application under your own domain.	BC-CP-CF-SEC-DOM	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal)	Yes	Available
Custom Domain Service	Configure and expose your application under your own domain.	BC-CP-CF-SEC-DOM	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Japan (Tokyo) Europe (Netherlands) US West (WA) US East (VA) Australia (Sydney) Singapore	Yes	Available
Custom Domain Service	Configure and expose your application under your own domain.	BC-CP-CF-SEC-DOM	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available
Custom Domain Service	Configure and expose your application under your own domain.	BC-CP-CF-SEC-DOM	Cloud Foundry	Extension Suite - Development Efficiency	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Data Attribute Recommendation	Apply machine learning to match and classify data records automatically.	CA-ML-DAR	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt)	Yes	Available
Data Enrichment	Create or enrich master data using trusted third-party data.	LOD-MDM-DE	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA) Singapore	Yes	Available
Data Quality Services	Embed data quality services to validate addresses and enrich with geocodes.	EIM-DQM-SVS	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Java Debugging	Debug your Java application even through networks with high latency.	BC-JVM	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Destination	Retrieve information about destinations in the Cloud Foundry environment.	BC-CP-DEST-CF	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal) US East (VA)	Yes	Available
Destination	Retrieve information about destinations in the Cloud Foundry environment.	BC-CP-DEST-CF	Cloud Foundry	Integration Suite	Azure	Japan (Tokyo) Europe (Netherlands) US West (WA) US East (VA) Singapore Australia (Sydney)	Yes	Available
Destination	Retrieve information about destinations in the Cloud Foundry environment.	BC-CP-DEST-CF	Cloud Foundry	Integration Suite	GCP	US Central (IA)	Yes	Available
Destination	Retrieve information about destinations in the Cloud Foundry environment.	BC-CP-DEST-CF	Cloud Foundry	Integration Suite	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Mobile Services, users	Build and run mobile apps for B2E and B2B use cases.	MOB-CLD-OPS	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Brazil (São Paulo) Canada (Toronto) KSA (Riyadh) UAE (Dubai) Russia (Moscow)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Mobile Services, users	Build and run mobile apps for B2E and B2B use cases.	MOB-CLD-OPS	Cloud Foundry	Extension Suite - Digital Experience	AWS	Australia (Sydney) Brazil (São Paulo) Japan (Tokyo) US East (VA) Europe (Frankfurt) Singapore Canada (Montreal) South Korea (Seoul) Europe (Frankfurt) EU Access	Yes	Available
Mobile Services, users	Build and run mobile apps for B2E and B2B use cases.	MOB-CLD-OPS	Cloud Foundry	Extension Suite - Digital Experience	Azure	Europe (Netherlands) US West (WA) Japan (Tokyo) US East (VA) Singapore	Yes	Available
Mobile Services, users	Build and run mobile apps for B2E and B2B use cases.	MOB-CLD-OPS	Cloud Foundry	Extension Suite - Digital Experience	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Mobile Services, consumers	Build and run mobile apps for B2C use cases.	MOB-CLD-OPS	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Brazil (São Paulo) Canada (Toronto) KSA (Riyadh) UAE (Dubai) Russia (Moscow)		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Mobile Services, consumers	Build and run mobile apps for B2C use cases.	MOB-CLD-OPS	Cloud Foundry	Extension Suite - Digital Experience	AWS	Australia (Sydney) Brazil (São Paulo) Japan (Tokyo) US East (VA) Europe (Frankfurt) Singapore Canada (Montreal) South Korea (Seoul) Europe (Frankfurt) EU Access	Yes	Available
Mobile Services, consumers	Build and run mobile apps for B2C use cases.	MOB-CLD-OPS	Cloud Foundry	Extension Suite - Digital Experience	Azure	Europe (Netherlands) US West (WA) Japan (Tokyo) US East (VA) Singapore	Yes	Available
Mobile Services, consumers	Build and run mobile apps for B2C use cases.	MOB-CLD-OPS	Cloud Foundry	Extension Suite - Digital Experience	Alibaba	China (Shanghai)**	Yes	Available
Document Classification	Classify business documents automatically using machine learning.	CA-ML-BDP	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Document Information Extraction	Automate your document information extraction processes.	CA-ML-BDP	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access Japan (Tokyo) US East (VA)	Yes	Available
Document Management Service, Integration Option	Provide API and UI based document management capabilities to your business applications.	BC-CP-CF-SDM	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt) US East (VA) Japan (Tokyo) Australia (Sydney) Singapore Brazil (São Paulo) Canada (Montreal) Europe (Frankfurt) EU Access South Korea (Seoul)	Yes	Available
Document Management Service, Integration Option	Provide API and UI based document management capabilities to your business applications.	BC-CP-CF-SDM	Cloud Foundry	Extension Suite - Digital Experience	Azure	Europe (Netherlands) US West (WA) US East (VA) Japan (Tokyo) Singapore	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Document Service	Store and manage your documents.	BC-NEO-ECM-DS	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Dynatrace Agent Activation	Connect your Java applications to a Dynatrace SaaS monitoring environment.	BC-NEO-MON-APM	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Brazil (São Paulo) Canada (Toronto) UAE (Dubai) KSA (Riyadh) Russia (Moscow)	Yes	Available
Enhanced Disaster Recovery	Restore your cloud production environment with minimal data loss.	BC-NEO-DR	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* US East (Ashburn) US West (Chandler) US East (Sterling) Australia (Sydney)		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Event Mesh	Connect applications, services and systems across different landscapes.	BC-CP-CF-MES	Cloud Foundry	Integration Suite	AWS	US East (VA) Australia (Sydney) Singapore Brazil (São Paulo) Japan (Tokyo) Europe (Frankfurt) Canada (Montreal)	Yes	Available
Event Mesh	Connect applications, services and systems across different landscapes.	BC-CP-CF-MES	Cloud Foundry	Integration Suite	Azure	Japan (Tokyo) Europe (Netherlands) US West (WA) US East (VA) Singapore	Yes	Available
Event Mesh	Connect applications, services and systems across different landscapes.	BC-CP-CF-MES	Cloud Foundry	Integration Suite	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Extension Factory, serverless runtime	Allows you to create, manage, configure extensions on SAP Cloud Platform	BC-CP-XF-SRT	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Brazil (São Paulo) Canada (Montreal)	Yes	Available
Extension Factory, serverless runtime	Allows you to create, manage, configure extensions on SAP Cloud Platform	BC-CP-XF-SRT	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Japan (Tokyo) Singapore	Yes	Available
Feature Flags	Control the rollout of new features.	BC-CP-CF-FEATUR-EFLG	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access Australia (Sydney) US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Feature Flags	Control the rollout of new features.	BC-CP-CF-FEATUR-EFLG	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Japan (Tokyo) Europe (Netherlands) US West (WA) US East (VA) Australia (Sydney) Singapore	Yes	Available
Feature Flags	Control the rollout of new features.	BC-CP-CF-FEATUR-EFLG	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available
Feature Flags	Control the rollout of new features.	BC-CP-CF-FEATUR-EFLG	Cloud Foundry	Extension Suite - Development Efficiency	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Fiori Cloud	Revamp your user experience with SAP Fiori on SAP Cloud Platform.	EP-CPP-NEO-OPS	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* Europe (Frankfurt) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) Europe (Amsterdam) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Fiori Mobile	Optimize, build, manage, and monitor SAP Fiori apps on mobile devices.	MOB-FM	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* US East (Ashburn) US West (Chandler) Australia (Sydney) Japan (Tokyo) US East (Sterling) Canada (Toronto) Russia (Moscow) Brazil (São Paulo)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Forms Service by Adobe	Generate print and interactive forms using Adobe Document Services.	BC-SRV-FP-NEO	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* US East (Ashburn) US West (Chandler) US East (Sterling) Australia (Sydney) Japan (Tokyo) Russia (Moscow) Brazil (São Paulo) Canada (Toronto) UAE (Dubai) Europe (Amsterdam) KSA (Riyadh) Europe (Frankfurt) US West (Colorado Springs)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Gamification	Introduce gamification concepts into your applications.	BC-NEO-SVC-GAM	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Git Service	Store and version source code in Git repositories.	BC-NEO-GIT	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
HTML5 Application Repository Service	Enables storage of HTML5 applications and provides runtime environment for HTML5 applications.	BC-CP-CF-HTML5	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Australia (Sydney) Singapore South Korea (Seoul) Brazil (São Paulo) Canada (Montreal) Japan (Tokyo)	Yes	Available
HTML5 Application Repository Service	Enables storage of HTML5 applications and provides runtime environment for HTML5 applications.	BC-CP-CF-HTML5	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Japan (Tokyo) Europe (Netherlands) US West (WA) US East (VA) Singapore	Yes	Available
HTML5 Application Repository Service	Enables storage of HTML5 applications and provides runtime environment for HTML5 applications.	BC-CP-CF-HTML5	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
HTML5 Application Repository Service	Enables storage of HTML5 applications and provides runtime environment for HTML5 applications.	BC-CP-CF-HTML5	Cloud Foundry	Extension Suite - Development Efficiency	Alibaba	China (Shanghai)**	Yes	Available
Hyperledger Fabric	Create Hyperledger Fabric nodes and connect them to a blockchain network.	BC-BCS-HL	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) US East (VA)	Yes	Available
Identity Authentication - Additional Tenant	Secure authentication and single sign-on for users in the cloud.	BC-IAM-IDS		Extension Suite - Development Efficiency	SAP	This service does not run on standard SAP Cloud Platform regions. Check out this note for further details.		Available
Identity Authentication	Secure authentication and single sign-on for users in the cloud.	BC-IAM-IDS		Extension Suite - Development Efficiency	SAP	This service does not run on standard SAP Cloud Platform regions. Check out this note for further details.		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Identity Provisioning	Manage identity lifecycle processes for cloud and on-premise systems.	BC-IAM-IPS	Neo	Extension Suite - Development Efficiency	SAP	UAE (Dubai) Australia (Sydney) Brazil (São Paulo) Canada (Toronto) Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) Japan (Tokyo) Russia (Moscow) KSA (Riyadh) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs)	Yes	Available
Intelligent Product Design	Accelerate product innovation with instant collaboration and live product intelligence.	PLM-DC	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt)		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Intelligent Product Design	Accelerate product innovation with instant collaboration and live product intelligence.	PLM-DC	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt)	Yes	Available
Internet of Things	Develop, customize, and operate IoT business applications in the cloud.	BC-NEO-SVC-IOT	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) US East (VA)	Yes	Available
Invoice Object Recommendation	Recommendation of the G/L accounts using machine learning.	CA-ML-AR-GL	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt)	Yes	Available
Peppol Exchange	Meet compliance requirements by exchanging documents with the Peppol network.	LOD-LH-DCS-PAP	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)*		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Java Server	Develop and run Java Web applications on SAP Cloud Platform Neo Environment.	BC-NEO-RT-JAV	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Job Scheduling Service	Define and manage your jobs or Cloud Foundry tasks that run on one-time or recurring schedules.	BC-XS-SRV-JBS	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal)	Yes	Available
Job Scheduling Service	Define and manage your jobs or Cloud Foundry tasks that run on one-time or recurring schedules.	BC-XS-SRV-JBS	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Japan (Tokyo) Europe (Netherlands) US West (WA) US East (VA) Australia (Sydney) Singapore	Yes	Available
Job Scheduling Service	Define and manage your jobs or Cloud Foundry tasks that run on one-time or recurring schedules.	BC-XS-SRV-JBS	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Job Scheduling Service	Define and manage your jobs or Cloud Foundry tasks that run on one-time or recurring schedules.	BC-XS-SRV-JBS	Cloud Foundry	Extension Suite - Development Efficiency	Alibaba	China (Shanghai)**	Yes	Available
Keystore Service	Manage cryptographic keys and certificates.	BC-NEO-SEC-CPG	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Kyma runtime	Extend SAP solutions using cloud-native microservices and serverless Functions.	BC-CP-XF-KYMA	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA)	Yes	Available
Java Apps Lifecycle Management	Manage the lifecycle of Java applications by using a REST API.	BC-NEO-INFR	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Live Link 365 for Authentication	Generate and validate configurable one-time PINs or verification codes.	BC-IAM-IDS		Extension Suite - Digital Experience	SAP	This service does not run on standard SAP Cloud Platform regions. Check out this note for further details.		Available
SAP Live Link 365 for SMS	Send and receive SMSs globally via REST APIs. View traffic logs and analytics.	CEC-DI-INE		Extension Suite - Digital Experience	SAP	This service does not run on standard SAP Cloud Platform regions. Check out this note for further details.		Available
Rapid Application Development by Mendix	Develop SAP business applications with a low-code, graphical toolset.	XX-PART-MDX-RAD		Extension Suite - Digital Experience	SAP	This service does not run on standard SAP Cloud Platform regions. Check out this note for further details.		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
MongoDB	Implement a NoSQL document store.	BC-NEO-BS-MONGO	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal)	Yes	Available
MongoDB	Implement a NoSQL document store.	BC-NEO-BS-MONGO	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo) Australia (Sydney)	Yes	Available
MongoDB	Implement a NoSQL document store.	BC-NEO-BS-MONGO	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Monitoring	Monitor Java and SAP HANA XS applications, and databases.	BC-NEO-MON	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available
MultiChain	Create MultiChain nodes and connect them to a blockchain network.	BC-BCS-MC	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) US East (VA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
OAuth 2.0	Protect applications and APIs with OAuth 2.0.	BC-NEO-SEC-IAM	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Object Store	Supports storage and management of unstructured data (files, BLOBs).	BC-CP-CF-OSAAS	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal) US East (VA)	Yes	Available
Object Store	Supports storage and management of unstructured data (files, BLOBs).	BC-CP-CF-OSAAS	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Japan (Tokyo) Europe (Netherlands) US West (WA) US East (VA) Australia (Sydney) Singapore	Yes	Available
Object Store	Supports storage and management of unstructured data (files, BLOBs).	BC-CP-CF-OSAAS	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
OData Provisioning	Access data in SAP Business Suite using OData services.	OPU-GW-OD-FW	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) KSA (Riyadh) Canada (Toronto) Russia (Moscow) Brazil (São Paulo)	Yes	Available
OData Provisioning	Access data in SAP Business Suite using OData services.	OPU-GW-OD-FW	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA)	Yes	Available
OData Provisioning	Access data in SAP Business Suite using OData services.	OPU-GW-OD-FW	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) Japan (Tokyo)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Open Connectors	Simplify integration via APIs	LOD-OCN-OPS	Neo	Integration Suite	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US West (Chandler) US West (Colorado Springs)	Yes	Available
Open Connectors	Simplify integration via APIs	LOD-OCN-OPS	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) US East (VA) Australia (Sydney) Singapore Canada (Montreal) Brazil (São Paulo) Japan (Tokyo)	Yes	Available
Open Connectors	Simplify integration via APIs	LOD-OCN-OPS	Cloud Foundry	Integration Suite	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo) Australia (Sydney)	Yes	Available
SAP S/4HANA Cloud for Intelligent Product Selection	Enable an intelligent customer experience for complex configurable products	LOD-PCI	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Platform Identity Provider	Use your user base from the identity authentication tenant for admin tasks.	BC-NEO-SEC-IAM	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Portal	Create role based, multi-channel sites to access business apps and content.	EP-CPP-NEO-OPS	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* Europe (Frankfurt) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) Europe (Amsterdam) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Portal	Create role based, multi-channel sites to access business apps and content.	EP-CPP-CF-OPS	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt) US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore Canada (Montreal) Europe (Frankfurt) EU Access South Korea (Seoul)	Yes	Available
Portal	Create role based, multi-channel sites to access business apps and content.	EP-CPP-CF-OPS	Cloud Foundry	Extension Suite - Digital Experience	Azure	Singapore US West (WA) US East (VA) Europe (Netherlands) Japan (Tokyo) Australia (Sydney)	Yes	Available
Portal	Create role based, multi-channel sites to access business apps and content.	EP-CPP-CF-OPS	Cloud Foundry	Extension Suite - Digital Experience	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Portal	Create role based, multi-channel sites to access business apps and content.	EP-CPP-CF-OPS	Cloud Foundry	Extension Suite - Digital Experience	Alibaba	China (Shanghai)**	Yes	Available
PostgreSQL	Consume an object-relational database with PostgreSQL.	BC-NEO-BS-POSTGRES	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal)	Yes	Available
PostgreSQL	Consume an object-relational database with PostgreSQL.	BC-NEO-BS-POSTGRES	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Australia (Sydney) Singapore Japan (Tokyo)	Yes	Available
PostgreSQL	Consume an object-relational database with PostgreSQL.	BC-NEO-BS-POSTGRES	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Pricing service	Calculate prices for configurable- and non-configurable products	LOD-CPS	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA) Singapore	Yes	Available
Print Service	Manage print queues, connect print clients and monitor print status	BC-CCM-PRN-OM-SCP	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt)	Yes	Available
Process Visibility	Cloud Offering with End-to-End visibility on Business Processes	LOD-BPM-VIS	Cloud Foundry	Extension Suite - Digital Process Automation	AWS	Europe (Frankfurt) US East (VA) Australia (Sydney) Singapore Japan (Tokyo) Brazil (São Paulo) South Korea (Seoul) Europe (Frankfurt) EU Access Canada (Montreal)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Process Visibility	Cloud Offering with End-to-End visibility on Business Processes	LOD-BPM-VIS	Cloud Foundry	Extension Suite - Digital Process Automation	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo) Australia (Sydney)	Yes	Available
Process Visibility	Cloud Offering with End-to-End visibility on Business Processes	LOD-BPM-VIS	Cloud Foundry	Extension Suite - Digital Process Automation	Alibaba	China (Shanghai)**	Yes	Available
Variant Configuration service	Configure your SAP ERP or SAP S/4HANA products interactively in the cloud	LOD-CPS	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA) Singapore	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Java Profiling	Profile and analyze your Java applications.	BC-JVM	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available
Quorum	Create Quorum nodes and connect them to a blockchain network.	BC-BCS-QRM	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) US East (VA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
RabbitMQ	Get robust asynchronous messaging between applications.	BC-NEO-BS-RAB-BITMQ	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal)	Yes	Available
RabbitMQ	Get robust asynchronous messaging between applications.	BC-NEO-BS-RAB-BITMQ	Cloud Foundry	Integration Suite	Azure	Europe (Netherlands) US West (WA) US East (VA) Australia (Sydney) Singapore Japan (Tokyo)	Yes	Available
RabbitMQ	Get robust asynchronous messaging between applications.	BC-NEO-BS-RAB-BITMQ	Cloud Foundry	Integration Suite	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Redis	Implement an in-memory caching layer with Redis.	BC-NEO-BS-REDIS	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal)	Yes	Available
Redis	Implement an in-memory caching layer with Redis.	BC-NEO-BS-REDIS	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Australia (Sydney) Singapore Japan (Tokyo)	Yes	Available
Redis	Implement an in-memory caching layer with Redis.	BC-NEO-BS-REDIS	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Remote Data Sync	Synchronize data between remote databases and a cloud SAP HANA database.	BC-NEO-CON	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) US East (Ashburn) US West (Chandler) US East (Sterling) Australia (Sydney) Japan (Tokyo)	Yes	Available
Data Retention Manager	Manage retention and residence rules to block or delete personal data.	LOD-GDP-RM	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA) Australia (Sydney)	Yes	Available
Data Retention Manager	Manage retention and residence rules to block or delete personal data.	LOD-GDP-RM	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Business Application Studio	Develop, debug, test, and deploy SAP business applications.	CA-BAS	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA) Australia (Sydney) Canada (Montreal) Japan (Tokyo) Brazil (São Paulo) Singapore Europe (Frankfurt) EU Access South Korea (Seoul)	Yes	Available
SAP Business Application Studio	Develop, debug, test, and deploy SAP business applications.	CA-BAS	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Japan (Tokyo) Singapore	Yes	Available
SAP Business Application Studio	Develop, debug, test, and deploy SAP business applications.	CA-BAS	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available
SAP Business Application Studio	Develop, debug, test, and deploy SAP business applications.	CA-BAS	Cloud Foundry	Extension Suite - Development Efficiency	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Document Management Service, Application Option	Organize your documents with ready-to-use document management capabilities.	BC-CP-CF-SDM	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt) US East (VA) Japan (Tokyo) Australia (Sydney) Singapore Brazil (São Paulo) Canada (Montreal) Europe (Frankfurt) EU Access South Korea (Seoul)	Yes	Available
Document Management Service, Application Option	Organize your documents with ready-to-use document management capabilities.	BC-CP-CF-SDM	Cloud Foundry	Extension Suite - Digital Experience	Azure	Europe (Netherlands) US West (WA) US East (VA) Japan (Tokyo) Singapore	Yes	Available
Workflow Management	Digitize workflows, manage decisions and gain end-to-end process visibility	LOD-BPM-PFS	Cloud Foundry	Extension Suite - Digital Process Automation	AWS	Europe (Frankfurt) Australia (Sydney) US East (VA) Singapore Japan (Tokyo) Brazil (São Paulo) South Korea (Seoul) Europe (Frankfurt) EU Access Canada (Montreal)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Workflow Management	Digitize workflows, manage decisions and gain end-to-end process visibility	LOD-BPM-PFS	Cloud Foundry	Extension Suite - Digital Process Automation	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo) Australia (Sydney)	Yes	Available
Workflow Management	Digitize workflows, manage decisions and gain end-to-end process visibility	LOD-BPM-PFS	Cloud Foundry	Extension Suite - Digital Process Automation	Alibaba	China (Shanghai)**	Yes	Available
SAP Analytics Cloud, embedded edition	Analyze data via live connection to your business application's SAP HANA database.	LOD-ANA-OEM-CP	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt) US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore	Yes	Available
SAP API Business Hub	Discover, explore and test the APIs offered by SAP.	LOD-CHB		Integration Suite	SAP	This service does not run on standard SAP Cloud Platform regions. Check out this note for further details.		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP ASE Service	Create and consume SAP ASE databases.	BC-NEO-PERS	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Analytics Cloud	Carry out business intelligence, planning, and predictive analysis tasks.	LOD-ANA	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* US East (Ashburn) US West (Chandler) US East (Sterling) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Brazil (São Paulo) Europe (Frankfurt) UAE (Dubai) KSA (Riyadh)		Available
SAP Analytics Cloud	Carry out business intelligence, planning, and predictive analysis tasks.	LOD-ANA	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt) US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal)	Yes	Available
SAP Analytics Cloud	Carry out business intelligence, planning, and predictive analysis tasks.	LOD-ANA	Cloud Foundry	Extension Suite - Digital Experience	Azure	US West (WA) Europe (Netherlands)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Analytics Cloud	Carry out business intelligence, planning, and predictive analysis tasks.	LOD-ANA	Cloud Foundry	Extension Suite - Digital Experience	Alibaba	China (Shanghai)**	Yes	Available
Cloud Foundry Runtime	Operate polyglot cloud applications in Cloud Foundry.	BC-CP-CF	Cloud Foundry	Extension Suite - Development Efficiency	AWS	US East (VA) Europe (Frankfurt) Europe (Frankfurt) EU Access Brazil (São Paulo) Japan (Tokyo) Canada (Montreal) Australia (Sydney) Singapore South Korea (Seoul)	Yes	Available
Cloud Foundry Runtime	Operate polyglot cloud applications in Cloud Foundry.	BC-CP-CF	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo) Australia (Sydney)	Yes	Available
Cloud Foundry Runtime	Operate polyglot cloud applications in Cloud Foundry.	BC-CP-CF	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Cloud Foundry Runtime	Operate polyglot cloud applications in Cloud Foundry.	BC-CP-CF	Cloud Foundry	Extension Suite - Development Efficiency	Alibaba	China (Shanghai)**	Yes	Available
Integration Suite	Integrate applications, services, and systems across landscapes.	LOD-HCI-PI	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) US East (VA) Brazil (São Paulo) Canada (Montreal) Australia (Sydney) Singapore Japan (Tokyo)	Yes	Available
Integration Suite	Integrate applications, services, and systems across landscapes.	LOD-HCI-PI	Cloud Foundry	Integration Suite	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo)	Yes	Available
Integration Suite	Integrate applications, services, and systems across landscapes.	LOD-HCI-PI	Cloud Foundry	Integration Suite	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Conversational AI	Build and deploy innovative chatbots using this comprehensive end-to-end platform.	CA-ML-CAI	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt)	Yes	Available
SAP Customer Order Sourcing	Calculate sourcing results based on your own sourcing strategies.	LOD-CID-OSR	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt)	Yes	Available
SAP Data Intelligence	Orchestrate, refine, enrich, apply intelligence on your governed data across your entire distributed data landscape.	CA-DI	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) US East (VA) Australia (Sydney) Japan (Tokyo)	Yes	Available
SAP Data Intelligence	Orchestrate, refine, enrich, apply intelligence on your governed data across your entire distributed data landscape.	CA-DI	Cloud Foundry	Integration Suite	Azure	Europe (Netherlands) US West (WA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Document Center	Use uniform standard-based file access and mobilize your business content.	BC-NEO-ECM-APP	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available
SAP Excise Tax Management	Help Excise Tax customers calculate, track, and comply with excise duty tax requirements in real time	LOD-ET-INT	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP HANA Cloud	A single gateway to all your data.	HAN-CLS-HC	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Australia (Sydney) Singapore Brazil (São Paulo) Canada (Montreal) Europe (Frankfurt) Japan (Tokyo) US East (VA)	Yes	Available
SAP HANA Cloud	A single gateway to all your data.	HAN-CLS-HC	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Singapore Europe (Netherlands) Japan (Tokyo) US West (WA) US East (VA)	Yes	Available
SAP HANA spatial services	SAP HANA spatial services provides a set of APIs for location-based services.	BC-CP-CF-HSS	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP HANA Service	Create and consume SAP HANA databases.	BC-NEO-PERS	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available
SAP HANA Service	Create and consume SAP HANA databases.	HAN-CLS-DB	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore South Korea (Seoul)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP HANA Service	Create and consume SAP HANA databases.	HAN-CLS-DB	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo) Australia (Sydney)	Yes	Available
SAP HANA Service	Create and consume SAP HANA databases.	HAN-CLS-DB	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available
SAP HANA Service	Create and consume SAP HANA databases.	HAN-CLS-DB	Cloud Foundry	Extension Suite - Development Efficiency	Alibaba	China (Shanghai)**	Yes	Available
SAP Intelligent Robotic Process Automation	Design, configure, and execute automation projects.	CA-ML-IPA	Cloud Foundry	Extension Suite - Digital Process Automation	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access Australia (Sydney) Japan (Tokyo) US East (VA)	Yes	Available
SAP Intelligent Robotic Process Automation	Design, configure, and execute automation projects.	CA-ML-IPA	Cloud Foundry	Extension Suite - Digital Process Automation	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP IoT	Put raw sensor data into business context and leverage it in analytical or transactional applications.	IOT-BSV-APB	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt) US East (VA)	Yes	Available
SAP IoT	Put raw sensor data into business context and leverage it in analytical or transactional applications.	IOT-BSV-APB	Cloud Foundry	Integration Suite	Azure	Europe (Netherlands)	Yes	Available
SAP IoT Connect 365	Simplify the complex connectivity, scalability, and management of IoT.	BC-NEO-SVC-IOT		Extension Suite - Development Efficiency	SAP	This service does not run on standard SAP Cloud Platform regions. Check out this note for further details.		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Launchpad Service	Simplify access to applications by establishing a central launchpad.	EP-CPP-CF	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt) US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore Canada (Montreal) Europe (Frankfurt) EU Access South Korea (Seoul)	Yes	Available
Launchpad Service	Simplify access to applications by establishing a central launchpad.	EP-CPP-CF	Cloud Foundry	Extension Suite - Digital Experience	Azure	Singapore US West (WA) US East (VA) Europe (Netherlands) Japan (Tokyo) Australia (Sydney)	Yes	Available
Launchpad Service	Simplify access to applications by establishing a central launchpad.	EP-CPP-CF	Cloud Foundry	Extension Suite - Digital Experience	Alibaba	China (Shanghai)**	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Leonardo ML Foundation	Infuse your applications with intelligent, easy-to-use services based on Machine Learning.	CA-ML-PLT	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Japan (Tokyo)	Yes	Available
SAP Leonardo ML Foundation	Infuse your applications with intelligent, easy-to-use services based on Machine Learning.	CA-ML-PLT	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available
Market Rates, Bring Your Own Rates	Upload market rates and download the same multiple times from different systems.	LOD-CBS-CS	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA)	Yes	Available
Market Rates, Refinitiv	Get daily and historical Exchange Rates and Interest Rates from Refinitiv.	LOD-CBS-CS	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt)	Yes	Available
SAP Omnichannel Promotion Pricing	Calculate effective sales prices by applying promotional rules.	LOD-CID-OPP	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Process Mining by Celonis, cloud edition	Mine data from your operational systems, visualize processes, and transform your business with AI.	XX-PART-CEL	Cloud Foundry	Extension Suite - Digital Process Automation	AWS	Europe (Frankfurt)	Yes	Available
SAP Procurement Intelligence	Automate and simplify your business processes by applying machine learning intelligence.	CA-ML-PA	Cloud Foundry	Integration Suite	AWS	Europe (Frankfurt)	Yes	Available
SAP RealSpend	Manage and track expenses, and compare them against your budget in real time.	LOD-MAP-RS	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt)		Available
SAP S/4HANA Cloud Extensibility	Connects extension applications running in a subaccount in SAP BTP to an SAP S/4HANA Cloud system.	BC-NEO-EXT-S4C	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore Canada (Montreal)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP S/4HANA Cloud Extensibility	Connects extension applications running in a subaccount in SAP BTP to an SAP S/4HANA Cloud system.	BC-NEO-EXT-S4C	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) Japan (Tokyo)	Yes	Available
SAP S/4HANA Cloud Extensibility	Connects extension applications running in a subaccount in SAP BTP to an SAP S/4HANA Cloud system.	BC-NEO-EXT-S4C	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available
SAP SuccessFactors Extensibility	Connects extension applications running in a subaccount in SAP BTP to an SAP SuccessFactors system.	BC-NEO-EXT-SF	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Brazil (São Paulo) Australia (Sydney) Europe (Frankfurt) US East (VA) Canada (Montreal) Singapore Japan (Tokyo)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP SuccessFactors Extensibility	Connects extension applications running in a subaccount in SAP BTP to an SAP SuccessFactors system.	BC-NEO-EXT-SF	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) Japan (Tokyo)	Yes	Available
SAP SuccessFactors Extensibility	Connects extension applications running in a subaccount in SAP BTP to an SAP SuccessFactors system.	BC-NEO-EXT-SF	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available
SAP Translation Hub	Translate UI texts and get suggestions for UI texts during development.	LOD-TH	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) Australia (Sydney) Japan (Tokyo) Europe (Frankfurt)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Web IDE Full-Stack	Create and extend SAP full-stack applications for browsers and mobile devices.	CA-WDE-PLFRM	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Brazil (São Paulo) Canada (Toronto) Russia (Moscow) UAE (Dubai) KSA (Riyadh)	Yes	Available
Service Manager	The central registry for service brokers and platforms in SAP BTP.	BC-NEO-SVCMGR	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA) Brazil (São Paulo) Japan (Tokyo) Australia (Sydney) Singapore Canada (Montreal)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Service Manager	The central registry for service brokers and platforms in SAP BTP.	BC-NEO-SVCMGR	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Japan (Tokyo) Singapore	Yes	Available
Service Manager	The central registry for service brokers and platforms in SAP BTP.	BC-NEO-SVCMGR	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available
Service Ticket Intelligence	Build a self-driven customer service powered by machine learning.	CA-ML-STI	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access	Yes	Available
Service Ticket Intelligence	Build a self-driven customer service powered by machine learning.	CA-ML-STI	Cloud Foundry	Extension Suite - Development Efficiency	GCP	US Central (IA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
SAP Smart Business Service	Expose KPIs and OPIs as SAP Fiori applications without the need to write any code.	CA-GTF-SB-HCP	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* Europe (Frankfurt) US East (Ashburn) US West (Chandler) US East (Sterling) Australia (Sydney) Japan (Tokyo) Europe (Frankfurt) US East (Sterling) Russia (Moscow) Canada (Toronto) Brazil (São Paulo)	Yes	Available
SAP Smart Business Service	Expose KPIs and OPIs as SAP Fiori applications without the need to write any code.	CA-GTF-SB-HCP	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt) US East (VA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Solutions Lifecycle Management	Deploy, subscribe and transport solutions using multi-target applications (MTAs).	BC-NEO-LCM-SRV	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) Australia (Sydney) Japan (Tokyo) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Streaming Analytics	Process continuous streams of event data in real time and act on the results.	HAN-SDS	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* US East (Ashburn) US West (Chandler) US East (Sterling) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Brazil (São Paulo) KSA (Riyadh) Russia (Moscow) Europe (Amsterdam) Europe (Frankfurt) UAE (Dubai)		Available
Tax Service	Determine and calculate indirect taxes to support tax compliance.	LOD-LH-TAX	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* US West (Chandler) US East (Sterling)	Yes	Available
Tax Service	Determine and calculate indirect taxes to support tax compliance.	LOD-LH-TAX	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) US East (VA)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Tax Service	Determine and calculate indirect taxes to support tax compliance.	LOD-LH-TAX	Cloud Foundry	Extension Suite - Development Efficiency	Azure	US West (WA)	Yes	Available
Cloud Transport Management	Provides programmatic access to Cloud Transport Management.	BC-CP-LCM-TMS	Cloud Foundry	Extension Suite - Development Efficiency	AWS	Europe (Frankfurt) Europe (Frankfurt) EU Access US East (VA) Australia (Sydney) Singapore South Korea (Seoul) Canada (Montreal) Brazil (São Paulo) Japan (Tokyo)	Yes	Available
Cloud Transport Management	Provides programmatic access to Cloud Transport Management.	BC-CP-LCM-TMS	Cloud Foundry	Extension Suite - Development Efficiency	Azure	Europe (Netherlands) US West (WA) US East (VA) Australia (Sydney) Singapore Japan (Tokyo)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
UI Theme Designer	Apply your corporate branding to applications based on SAPUI5 technology.	CA-UI2-THD	Neo	Extension Suite - Digital Experience	SAP	Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs) Australia (Sydney) Japan (Tokyo) Canada (Toronto) Russia (Moscow) Brazil (São Paulo) UAE (Dubai) KSA (Riyadh)	Yes	Available
UI Theme Designer	Apply your corporate branding to applications based on SAPUI5 technology.	CA-UI2-THD	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt) Brazil (São Paulo) Canada (Montreal) US East (VA) Australia (Sydney) Singapore Japan (Tokyo)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
UI Theme Designer	Apply your corporate branding to applications based on SAPUI5 technology.	CA-UI2-THD	Cloud Foundry	Extension Suite - Digital Experience	Azure	Europe (Netherlands) US West (WA) Japan (Tokyo) Singapore US East (VA)	Yes	Available
UI Theme Designer	Apply your corporate branding to applications based on SAPUI5 technology.	CA-UI2-THD	Cloud Foundry	Extension Suite - Digital Experience	GCP	US Central (IA)	Yes	Available
UI Theme Designer	Apply your corporate branding to applications based on SAPUI5 technology.	CA-UI2-THD	Cloud Foundry	Extension Suite - Digital Experience	Alibaba	China (Shanghai)**	Yes	Available
UI5 flexibility for key users	Add UI adaptation to your UI5 applications.	CA-UI5-FL-CLS	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt) US East (VA) Japan (Tokyo) Singapore South Korea (Seoul) Australia (Sydney) Canada (Montreal) Europe (Frankfurt) EU Access Brazil (São Paulo)	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
UI5 flexibility for key users	Add UI adaptation to your UI5 applications.	CA-UI5-FL-CLS	Cloud Foundry	Extension Suite - Digital Experience	Azure	Europe (Netherlands) Australia (Sydney) US West (WA) US East (VA) Singapore Japan (Tokyo)	Yes	Available
UI5 flexibility for key users	Add UI adaptation to your UI5 applications.	CA-UI5-FL-CLS	Cloud Foundry	Extension Suite - Digital Experience	GCP	US Central (IA)	Yes	Available
Virtual Machines	Virtualized hardware resources (CPU, RAM, disk space, installed OS) to install and maintain the Linux-based software.	BC-NEO-INFR-VM	Neo	Extension Suite - Development Efficiency	SAP	Europe (Rot)* Australia (Sydney) US East (Sterling)		Available
Web Analytics	Analyze usage of your websites and web applications.	CA-SWA	Cloud Foundry	Extension Suite - Digital Experience	AWS	Europe (Frankfurt) US East (VA) Australia (Sydney) Brazil (São Paulo) Japan (Tokyo) Singapore	Yes	Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Web Analytics	Analyze usage of your websites and web applications.	CA-SWA	Cloud Foundry	Extension Suite - Digital Experience	Azure	Europe (Netherlands) US West (WA) US East (VA) Singapore Japan (Tokyo)	Yes	Available
Workflow	Automate business processes using workflow technology.	LOD-BPM-WFS	Neo	Extension Suite - Digital Process Automation	SAP	UAE (Dubai) Australia (Sydney) Brazil (São Paulo) Canada (Toronto) Europe (Rot)* Europe (Frankfurt) Europe (Amsterdam) Japan (Tokyo) Russia (Moscow) KSA (Riyadh) US East (Ashburn) US West (Chandler) US East (Sterling) US West (Colorado Springs)		Available

Service Name	Short Description	Support Component	Environment	Capability	Infrastructure	Region	Available as Trial	Release Status
Workflow	Automate business processes using workflow technology.	LOD-BPM-WFS	Cloud Foundry	Extension Suite - Digital Process Automation	AWS	Australia (Sydney) Singapore South Korea (Seoul) Brazil (São Paulo) Canada (Montreal) Europe (Frankfurt) Europe (Frankfurt) EU Access Japan (Tokyo) US East (VA)	Yes	Available
Workflow	Automate business processes using workflow technology.	LOD-BPM-WFS	Cloud Foundry	Extension Suite - Digital Process Automation	Azure	Australia (Sydney) Singapore Europe (Netherlands) Japan (Tokyo) US West (WA) US East (VA)	Yes	Available
Workflow	Automate business processes using workflow technology.	LOD-BPM-WFS	Cloud Foundry	Extension Suite - Digital Process Automation	Alibaba	China (Shanghai)**	Yes	Available

Additional Components

Tools

Name	Support Component
SAP BTP Cockpit	BC-NEO-CPT [Feature Set A] BC-CP-CPT [Feature Set B] BC-CP-CF-CPT [Cloud Foundry environment UI]
SAP BTP Command Line Interface	BC-CP-TOOLS-CLI
SAP BTP Console Client	BC-NEO-CMDTOOL
SAP UI Development Toolkit for HTML5 (SAPUI5)	CA-WDE
SDK for SAP BTP, Neo Environment	BC-NEO-SDK
SAP HANA Cockpit 2.0	HAN-CLS-CPT

Software Logistics

Name	Support Component
Deployment (Neo Environment)	BC-NEO-DPL
Deployment (Cloud Foundry Environment)	BC-XS-SL-DS

Other

Name	Support Component
Infrastructure (Neo environment)	BC-NEO-IT-NW
SAP Cloud Management Service	BC-NEO-CIS
SAP Usage Data Management Service	BC-NEO-MET-REP
SAP Audit Log service	BC-NEO-AUDITLOG (Neo environment) BC-CP-CF-SEC-AUDITLG (Cloud Foundry environment)
SAP Streaming Analytics	HAN-SDS
SAP Cloud Application Programming Model (CAP)	BC-XS-CDX BC-XS-CDX-COR (Compiler and CDS Language) BC-XS-CDX-JAV (Java Runtime) BC-XS-CDX-NJS (Node.js Runtime) BC-XS-CDX-TLS (Tools, IDEs, Build, Deployment)

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.

THE BEST RUN SAP