API Manager
ADOBE® COLDFUSION (2016 release)

Contents

ColdFusion API Manager Overview

Overview

The most successful companies in today's digital era create and maintain unambiguous and stable interfaces for their business. Companies execute their strategies around extensive interfaces within and outside their enterprises. These interfaces are known as Application Programming Interfaces (APIs).

An API-centric approach encourages evolution and management of various interfaces between systems. In this approach, all systems are loosely coupled and allow associated services to have a wide range of uses. An API-centric approach is similar to Service Oriented Architecture (SOA) except that APIs deal with one-to-many architectures instead of one-one architecture of SOAs.

As a result, newer technologies, such as, REST web services make it easier to execute an enterprise API strategy.

Why API strategy

A well-planned API strategy helps businesses in executing complex software-driven strategies, such as:

- Better restructuring of internal software systems
- Reduction of maintenance costs
- Increase in agility and lean production processes
- Better customer engagement and retention

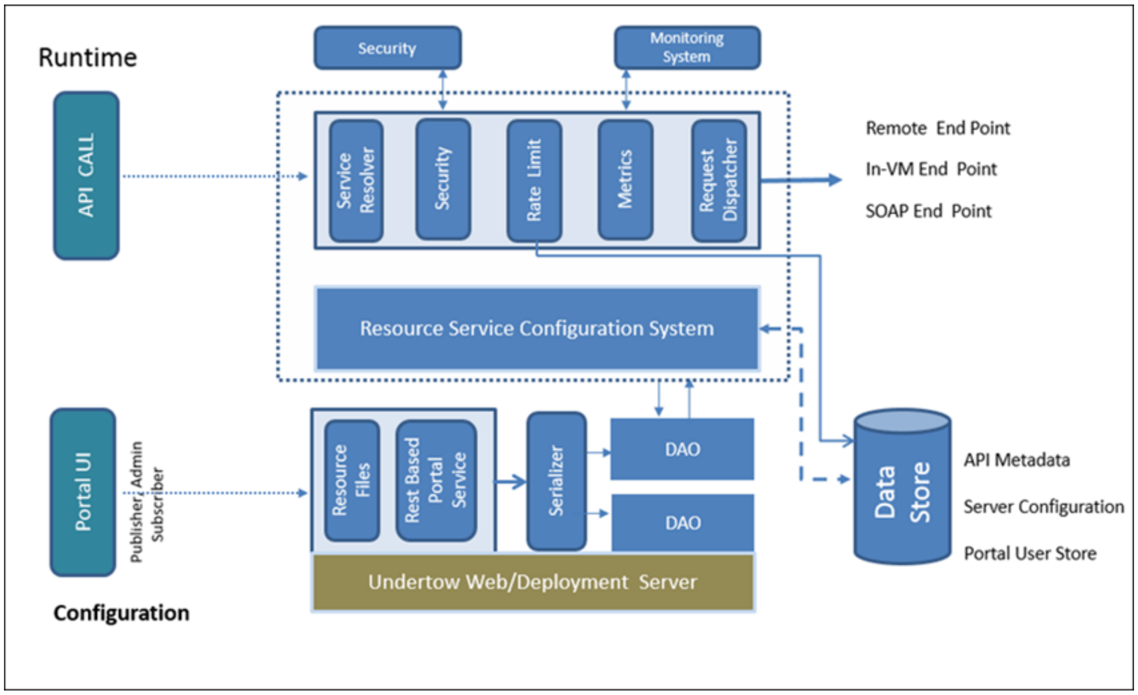
What is ColdFusion API Manager

ColdFusion API Manager helps you to create APIs that expose core functionalities of application and other backend systems. These APIs are then published and managed at runtime.

The API Manager includes the following functionalities, such as:

- **Access control:** The API Manager helps you to restrict certain APIs to trusted subscribers and enforce usage for each users based on roles.
- **Analytics:** You can track the usage of an API across applications, methods, and users.
- **Portal:** There are separate portals for an API administrator, publisher, and subscriber with custom workflows.

Architecture of API Manager



User roles in API Manager

The API Manager includes the following roles:

API Administrator-The API administrator manages and administers all core functions and properties of the API Manager.

API Publisher-The API publisher creates an API, lists resources of each API, views all subscribed users, and analyzes metrics of an API.

API Subscriber-The API subscriber can browse the subscribed APIs, select an API tier, create applications, and view notifications from the publisher.

Features of API Manager

The API Manager has the following features:

Feature	Description
Create an API	There are multiple ways to create an API, for example: • Manually create an API. • Import from Swagger. • Import internal Swagger from ColdFusion for ColdFusion-based REST services. • Convert a SOAP service into a REST service. • Import a SOAP API through a proxy service.
API Versioning	API versioning is essential when designing and building a REST API. As updates are made to your APIs, you want to make sure you can make interface changes without affecting the current usage of your API. In the API Manager, you can create multiple versions of the same API and manage them separately. You can later deprecate and retire a published version.
API Security	You can authenticate your API through the following security mechanisms: • Basic • API Keys • OAuth2 using SAML
Documentation	The publisher can provide the complete description of an API and a try-out tool for a subscriber to test the API.
Portal	In The API Manager, there are portals to configure, publish, and subscribe APIs, and view reports. The different types of portals are: • **Administrator portal:** You can configure server settings, SLAs, mail settings for notifications, and so on. • **Publisher portal:** You can create, publish, view, and manage APIs. • **Subscriber Portal:** You can subscribe, view, and test an API.

Analytic Dashboards	You can view metrics and API usage statistics in the API Manager via a dashboard. There are three different types of dashboard: • **Administrator dashboard:** You can analyze all the Publishers, their API's(with versioning) and their usage. You can take actions based on low performance or high usage of API's and much more. • **Publisher dashboard:** You can analyze overall metrics of multiple API's(with versioning) or you can compare your APIs (with versioning) based on factors like data consumed/generated and usage or request-time on time based graphs. You can take actions based on usage, performance or load(time based) of your APIs and much more. • **Subscriber dashboard:** You can analyze the API's(with versioning) you are subscribed to on multiple factors like data consumed/generated or usage or request time. Application/API based filtering is provided. You can figure out widely used API's or low performing API's. You can also take actions based on API's facing throttling limit with time graphs, and much more.
Metering and Throttling	The API Manager contains a scalable metering solution that works across all cluster nodes. The API Manager makes the asynchronous throttle calls based on the metrics data. The API Manager manages metering through a distributed in-memory database for real-time control.
SLAs and Tiers	The API Manager helps you create SLA tiers for your APIs by defining the throughput and enforcing approvals.
Monitoring and Analytics	The API Manager publishes all information, such as, publisher information, subscriber information, API resources, request and response payload, response status from end-point, response time, response format, and SLA plan to a monitoring server. This data is published on an analytics server within or outside a network. The administrator, or publisher, or consumer can generate reports and analyze the data.
Caching	The API Manager contains a caching system where an API response can be cached suitably. The caching can be enabled at subresource level if all the resources are not cacheable at the API level.
SOAP to REST conversion	The ColdFusion community develops and uses SOAP services extensively. The API Manager helps you to wrap the SOAP services and expose them as REST service externally. There is a wizard to help a publisher map the SOAP input parameters to that of the REST parameters.
SOAP Proxy	Import a SOAP API through a proxy service and map SOAP endpoints to proxy endpoints.
Clustering	The API Manager helps you to set up clusters using the IP of the machine that hosts the distributed data store services.

Connector	In the API Manager, the connector fetches all nodes through a shared file. A connector is by a ColdFusion server and the API Manager, running on different machines, at the same time.
Sandbox	In the API Manager, you can test your API before going to production by configuring a sandbox URL. At runtime, based on the API key, the API Manager calls the sandbox URL's endpoints and makes a request to the production server. There is also a failover server, which acts as a backup server in events when the primary endpoint server is unable to serve a request.
Notifications	You can notify users of certain events in the API Manager. For example, • Notify a subscriber if the subscriber approaches the end of a subscribed SLA. • Notify a publisher to approve a subscription if someone has made a request to subscribe to an API. • Notify all subscribers if a publisher downgrades the status of a published API to "Deprecate".

Installing ColdFusion API Manager

Windows

Linux

Mac OS X

The procedure for installing the API Manager in ColdFusion varies according to your platform (Microsoft Windows, Linux, or Apple Mac OS X).

Windows

The ColdFusion API Manager installation wizard navigates you through the various steps of the installation process through an intuitive user interface.

There is also an add-on installer that installs the API Manager data store and analytics server. You can install the API Manager in one machine and the data store and analytics server in separate machines.

System Requirements

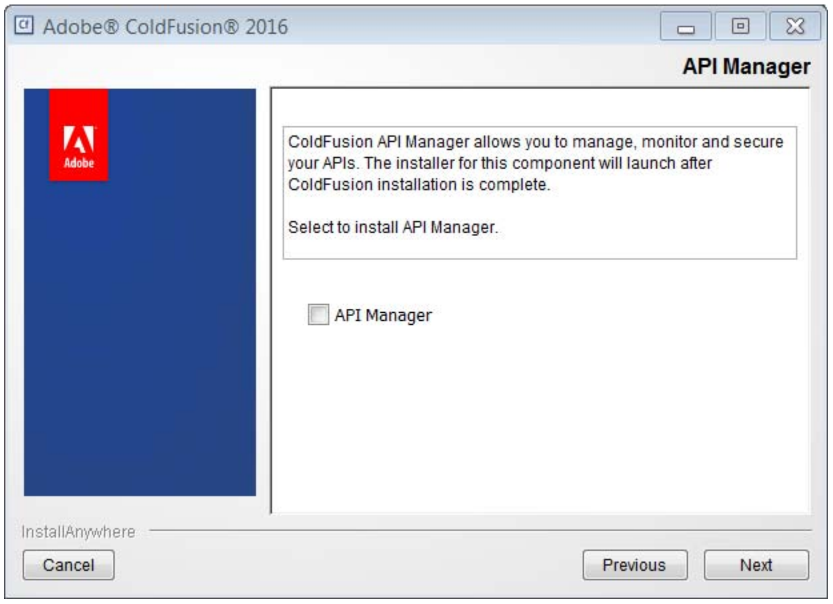
Windows

- Microsoft Windows 7, Windows 8, Windows 8.1, Windows 10, or Windows Server 2012 R2
- 2 GHz or faster processor
- 3-GB hard-disk space recommended
- Latest version of Google Chrome, Mozilla Firefox, or Internet Explorer browsers

To install the API Manager, you must have a **Enterprise-licensed** copy of Adobe ColdFusion (2016 release) Server installed in your system.

Installing the API Manager

The ColdFusion API Manager installer is available bundled with an enterprise licensed ColdFusion (2016 release) installer. If you run the installer, you can view the option to install ColdFusion API Manager, as shown below:



ColdFusion API Manager allows you to manage, monitor and secure your APIs. The installer for this component will launch after ColdFusion installation is complete.

Select to install API Manager.

☐ API Manager

Enable the **API Manager** check-box and continue with the installation process. At the end of ColdFusion installation, API Manager starts installing and you can follow the on-screen instructions.

This option is only available in Windows OS.

To install API Manager:

1. Enter the ColdFusion instance location. For example, <ColdFusion Installation Directory>/cfusion.
2. Enter the directory where you want to install API Manager.
3. Choose if you want to run API Manager in the same VM as ColdFusion. Select **Yes** or **No**.

If you select	Then
Yes	For In-VM, API Manager services will not get created assuming that you want to start API Manager along with Adobe ColdFusion (2016 release). After installation of API Manager, navigate to <ColdFusion Installation Directory>/cfusion/bin and enter cfstart.bat in the command prompt. The batch file launches API Manager along with ColdFusion.
No	After installing API Manager, services will be up and running. Open a browser and enter http://localhost:portnumber/admin to access Administrator portal and http://localhost:portnumber/portal to access Publisher/Subscriber portal.

***Note:** For ColdFusion installation in JEE and API manager in the same machine, if you want to run it in-VM, then add the flag -Dapim.home in the config file of JEE application server.*

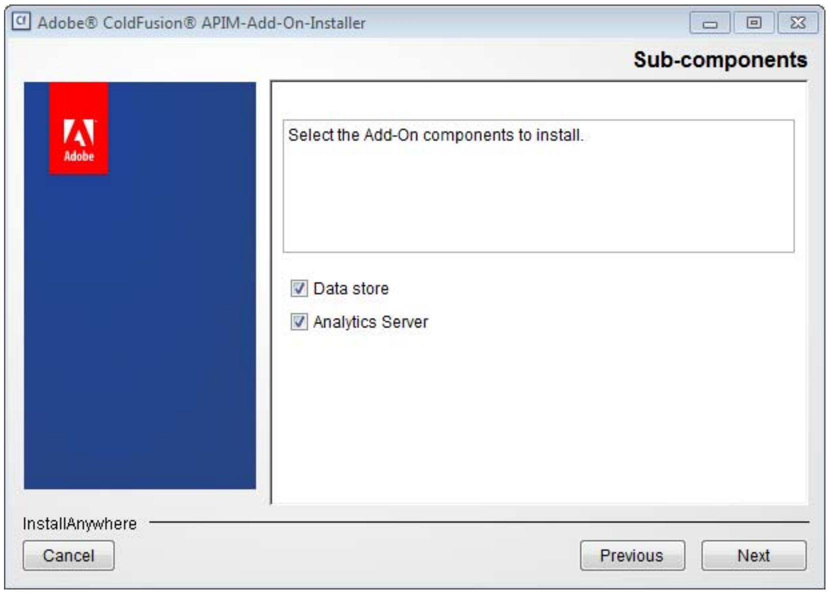
4 In the **Port** field, enter the port of the API Manager.

5 Select if you want to install **Analytics Server** or **Data Store** locally or remotely. Enter the ports for the sub components.

6 Enter the administrator credentials of API Manager.

Installing API Manager add-ons

You can also install the API Analytics Server and the API Data store using an add-on installer. The size of an add-on installer is less than the size of an API Manager installer. Choose both the options and follow the onscreen instructions. Enter the ports for the components and click **Install**.



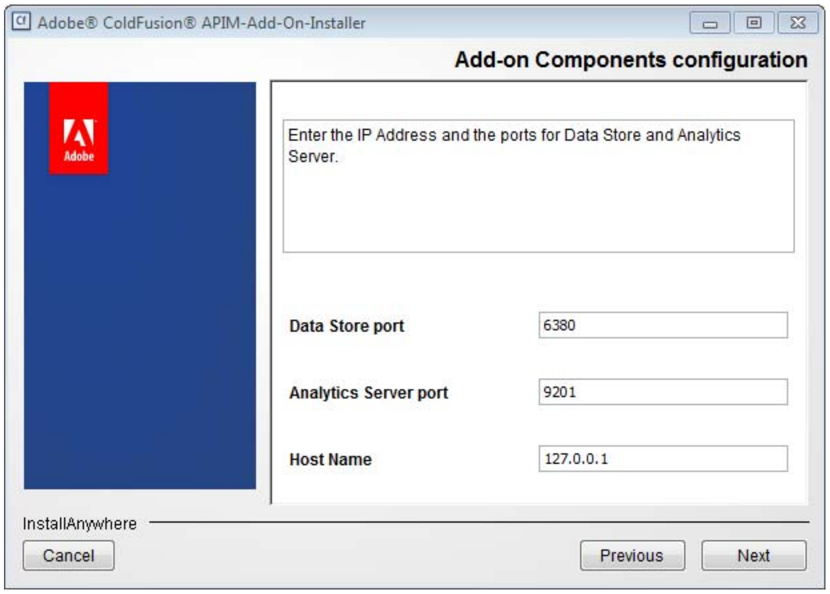
Adobe® ColdFusion® APIM-Add-On-Installer

Sub-components

Select the Add-On components to install.

☑ Data store

☑ Analytics Server


Adobe® ColdFusion® APIM-Add-On-Installer

Add-on Components configuration

Enter the IP Address and the ports for Data Store and Analytics Server.

Data Store port: 6380

Analytics Server port: 9201

Host Name: 127.0.0.1

Logging in to the Administrator portal

After the API Manager installs, log in to the Administrator portal.

The Administrator portal is a platform to manage and administer properties and functionality of the ColdFusion API Manager. Through the administrator portal, you can manage all published APIs, set up cluster and notifications, create user and user stores, and set API authentication policies.

To log in to the Administrator portal:

1 Open a browser and enter http://<locahost or IP address>:<port number>/admin

2 Enter the user name and password. Click Login.

Linux

After Adobe ColdFusion (2016 release) installs, the binary file is copied in the <ColdFusion install folder> directory.

You can use the installation steps on the following OS flavors:

- Solaris
- SUSE Linux
- RHEL
- Ubuntu

Starting Adobe API Manager on Linux

1 To install API Manager, enter ./ColdFusionAPIManager_2016_WWEJ_*.bin.

   The * represents the OS.

2 To start Data Store, navigate to <API_Manager_install_dir>/database/datastore/ and enter ./redis-server redis.conf.properties**.**

3 To start Analytics Server, navigate to <API_Manager_install_dir>/database/analytics/bin and enter ./elasticsearch.

4 To start Adobe API Manager, navigate to <API_Manager_install_dir>/bin and enter ./start.sh.

Mac OS X

To install API Manager in Mac OS, run ./ColdFusionAPIManager_2016_WWEJ_osx10.dmg and follow the on-screen instructions.

1 To start Data Store, navigate to <API_Manager_install_dir>/database/datastore/ and enter ./redis-server redis.conf.properties**.**

2 To start Analytics Server, navigate to <API_Manager_install_dir>/database/analytics/bin and enter./elasticsearch.

3 To start API Manager, open a browser and enter http:localhost:port/admin.

Administrator

The administrator portal is a platform to manage and administer properties and functionality of the ColdFusion API Manager. Through the administrator portal, you can manage all published APIs, and their functional and non-functional information.

As an administrator, you can perform the following tasks:

- Add and manage all users and their roles with respect to single or multiple published APIs.
- View and manage all SLA plans.
- View and manage tier plans of all published APIs.

Specifying server settings

Specifying security settings

Configuring SLAs

Viewing cluster information

Caching

ColdFusion discovery server

Configuring notifications

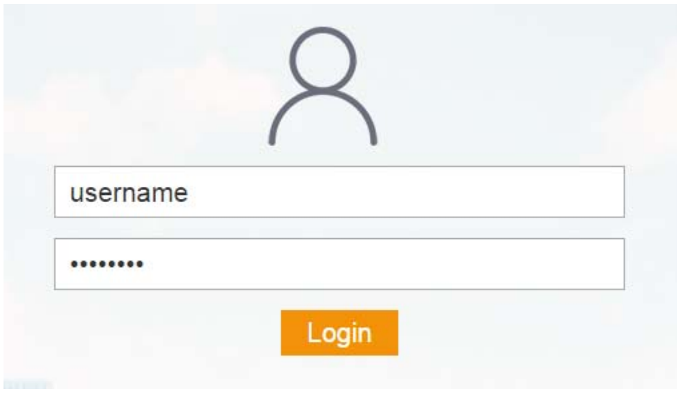
Configure log

Viewing the analytics dashboards

Logging in to the administrator portal

To log in to the administrator portal:

1. Enter the user name and password in the fields provided.
2. Click Login.



If you want to change the password, navigate to <API Manager Installation Directory>/conf and open the file password.properties. Change the password in the file. You can only change the default administrator password using the password.properties file.

Steps to reset admin password: 1. Edit the password.properties file at <apimanager_home/conf>with entries as: adminname=<User name while installing API Manager> adminpassword=<Your new preferred password> 2. Make readDefault to true in default_conf.xml. 3. Restart "ColdFusion 2016 API Manager" service.

You cannot delete or edit this account to remove administrator privileges.

You cannot change the password of other users except the default user.

Changing the header logo and title

You can change the logo and title on the header area of the administrator portal.

Navigate to the <API Manager Installation Directory>/wwwroot/portal/conf folder and open the file config.json. Modify the following properties:

```
{
 "headerTitle": "API Manager Portal",
 "adminHeaderTitle": "API Manager Administrator",
 "headerLogoPath": "images/CF-Logo.png"
}
```

Specifying server settings

You can configure the settings for the API Manager server from the following screens:

- General
- API Datastore
- API Analytics Server

General

On this page, you can set up an API Proxy or an API Portal to host your APIs.

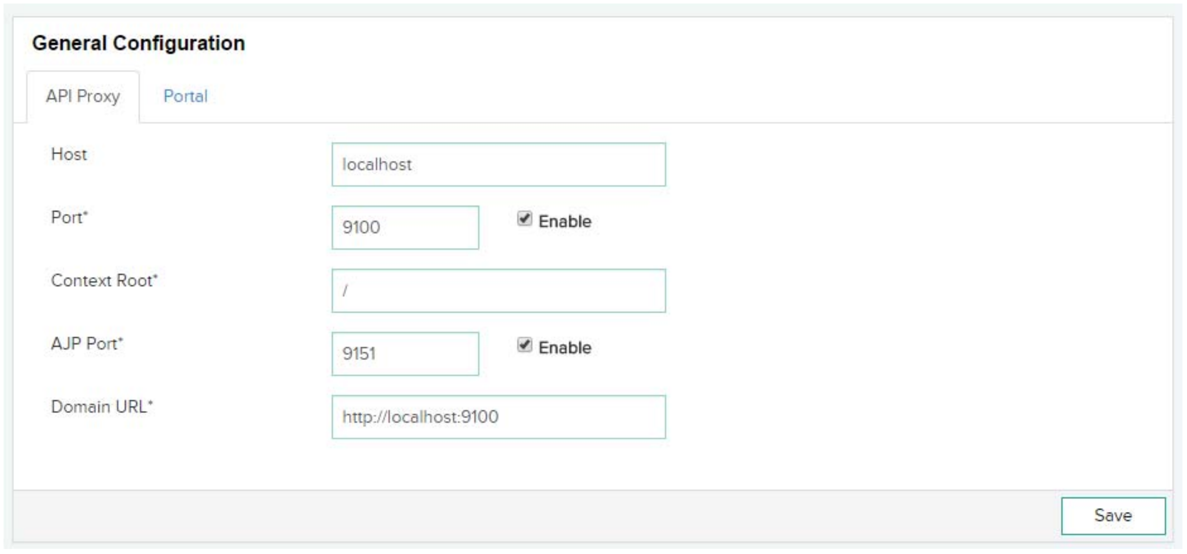
API Proxy

In an enterprise, you communicate with existing and new customers. Consumers use a range of devices from phones to desktops. One of the ways to interact with the devices is through an API. But, there are some concerns, for example, security, availability, and monitoring of the APIs. In such situations, you define an API proxy. An API proxy performs the following:

- Transports security
- Monitors SLAs, performance, and so on, of the API
- Defines access levels of an API

An API proxy makes your API independent from your back end services. Whenever you change your back end services, for example, changes in code, the application runs without interruptions.

Setting up an API Proxy


General Configuration

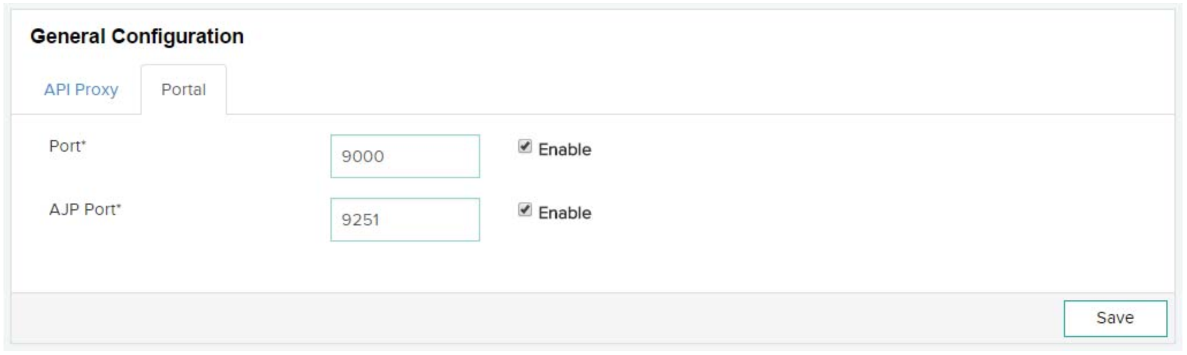
API Proxy | Portal

Host	localhost	
Port*	9100	✓ Enable
Context Root*	/	
AJP Port*	9151	✓ Enable
Domain URL*	http://localhost:9100	

1. In the Host field, enter the IP address. The API Proxy can be a host within the network or external to the network.
2. In the Port field, enter the port of the API Proxy. For example, http://localhost:5100, where 5100 is the port number.
3. To enable the port on the API Proxy when the proxy restarts (manually or scheduled), select the Enable check-box.
4. In the Context Root field, enter the name of the context root. A context root identifies an application on the server. For example, if your application's context root is mycontextroot, and the application is hosted on http://localhost:5100, you can access the application at http://localhost:5100/mycontextroot.
5. In the AJP Port field, enter the port number of the Apache JServ Protocol (AJP) connector. AJP allows Tomcat to communicate with an Apache web server.
6. To enable the AJP Port on the server, select the Enable check-box adjacent to the AJP Port field. To save the changes, click Save.
7. In the **Domain URL** field, enter the URL of the domain that hosts the API endpoints.

Setting up a Portal

Publishers use portals to create APIs for subscribers to consume. On a portal, a publisher can create the APIs and can restrict the APIs to certain nodes. Portals can run on multiple nodes. On the other hand, you can configure an API proxy on any node.


General Configuration

API Proxy | Portal

Port* 9000 ✓ Enable

AJP Port* 9251 ✓ Enable

Save

1 In the Port field, enter the port of the API Portal.

2 To enable the port on the API Portal when the proxy restarts (manually or scheduled), select the Enable check-box.

3 In the AJP Port field, enter the port number of the Apache JServ Protocol (AJP) connector. AJP allows Tomcat to communicate with an Apache web server.

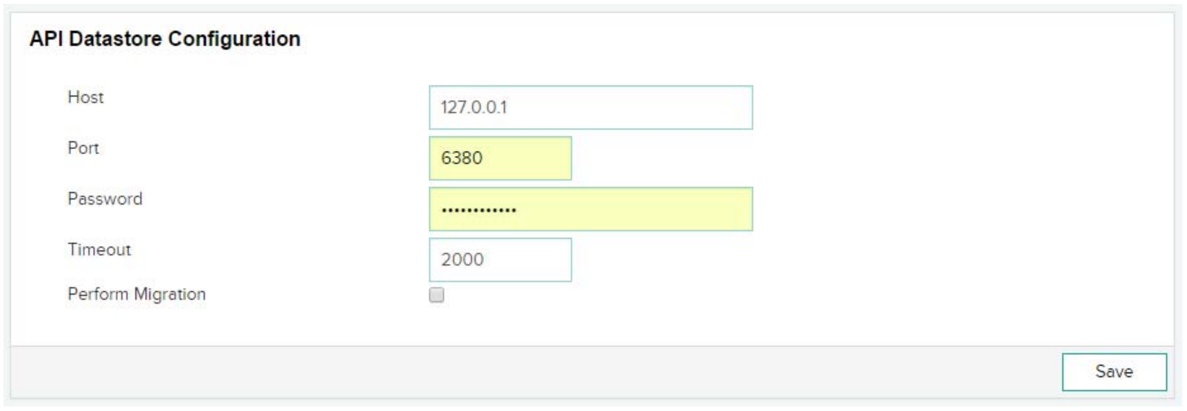
4 To enable the AJP Port on the server, select the Enable check-box adjacent to the AJP Port field. To save the changes, click Save.

API Datastore

The Datastore is a distributed configuration system and an in-memory cache used in the API Manager. The API Datastore has the following advantages:

- Distributed
- Fast
- Scalable
- Failsafe

Configuring datastore settings


API Datastore Configuration

Host	127.0.0.1
Port	6380
Password	············
Timeout	2000
Perform Migration	☐

1 In the Host field, enter the host name or IP address of the Data store. The Data store can be local or external to a network.

2 In the Port field, enter the port number of the Data Store.

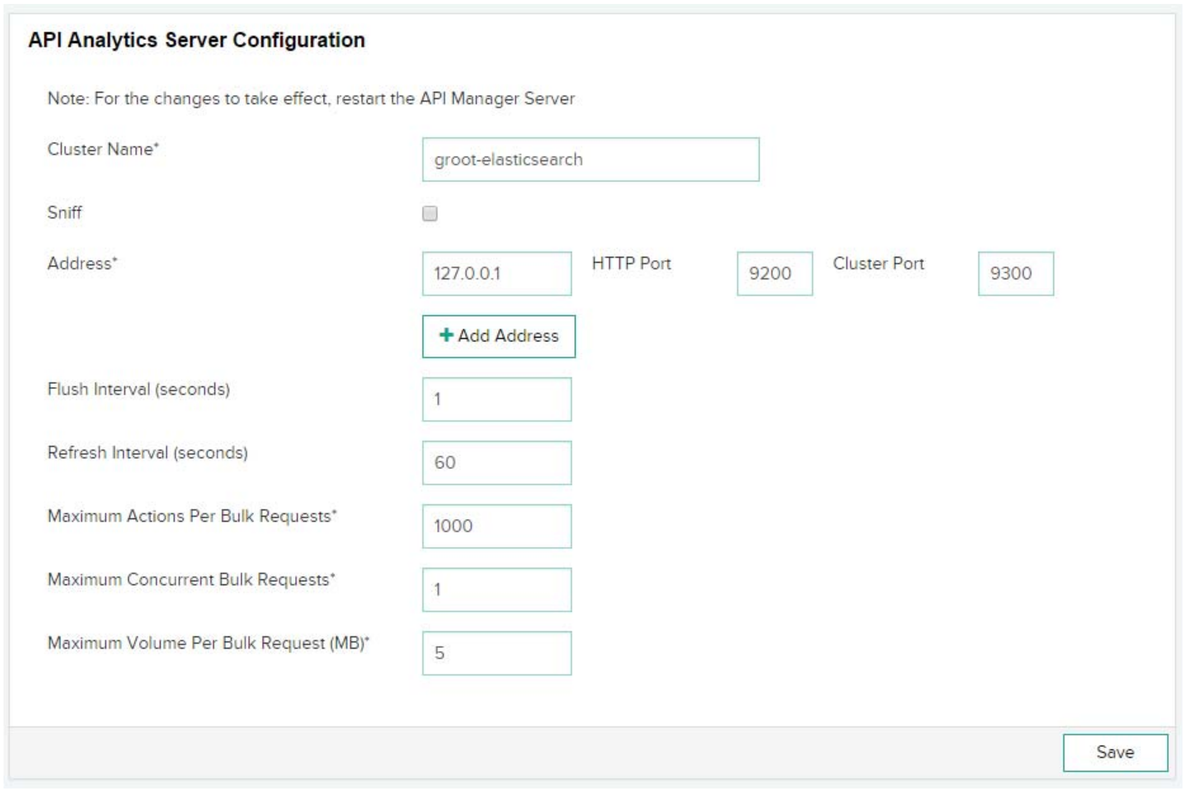
3 In the Password field, enter the password of the Data Store.

4 In the Timeout field, enter the timeout duration in seconds.

API Analytics Server

Configure the settings for the number of API requests for which to generate metrics. For more information, refer to API Manager-Metrics and Logging.

Configuring metric settings



API Analytics Server Configuration

Note: For the changes to take effect, restart the API Manager Server

Cluster Name*	groot-elasticsearch				
Sniff	☐				
Address*	127.0.0.1	HTTP Port	9200	Cluster Port	9300
Flush Interval (seconds)	1				
Refresh Interval (seconds)	60				
Maximum Actions Per Bulk Requests*	1000				
Maximum Concurrent Bulk Requests*	1				
Maximum Volume Per Bulk Request (MB)*	5				

1 In the **Cluster Name** field, enter the name of the cluster. The analytics of these requests are then published into a portal. For more details, refer to Setting up Cluster Support .

2 In Analytics Server configuration, the client can sniff the rest of the cluster, which adds nodes that the cluster can use. To enable the feature, select **Sniff**.

When you select Sniff and specify an Analytic Server address in the configuration, the API Manager finds out the other cluster members and load balances them in a round-robin fashion.

3 Specify all the node addresses along with the cluster port. To manually add the nodes, click **Add Address**.

4 Enter the port numbers in the **HTTP Port** and **Cluster Port** fields. The Cluster Port is the location where the API Manager sends the metric data to the Analytic Server. The analytic dashboard fetches the data from the HTTP Port.

5 In the **Flush Interval** field, enter the time interval (in seconds). This is the time after which the Analytic Server receives a specified number of API requests.

6 In the **Refresh Interval** field, enter the time interval (in seconds). This is the time after which the Analytic server refreshes one or more index.

7 In the Maximum Actions Per Bulk Request field, enter the number of API requests that are collectively sent to the analytic server for metrics calculation and visualization. If you enter 1000 in the field, no more than 1000 requests can be sent to the server.

The default value is 1000 actions per bulk request. This means that whenever, 1000 requests are received by the API Manager, a bulk request is dispatched to the Analytics server.

8 In the Maximum Concurrent Bulk Request field, enter the number of concurrent blocks of API requests that are sent to the analytic server. For example, if you enter 2 in the field, no more than two concurrent blocks of API requests can be sent to the server.

9 In the Maximum Volume Per Bulk Request, enter the size of the concurrent requests in this field. The size is calculated in MBs. For example, if you enter 5 in the field, the size of concurrent API requests cannot exceed 5 MB.

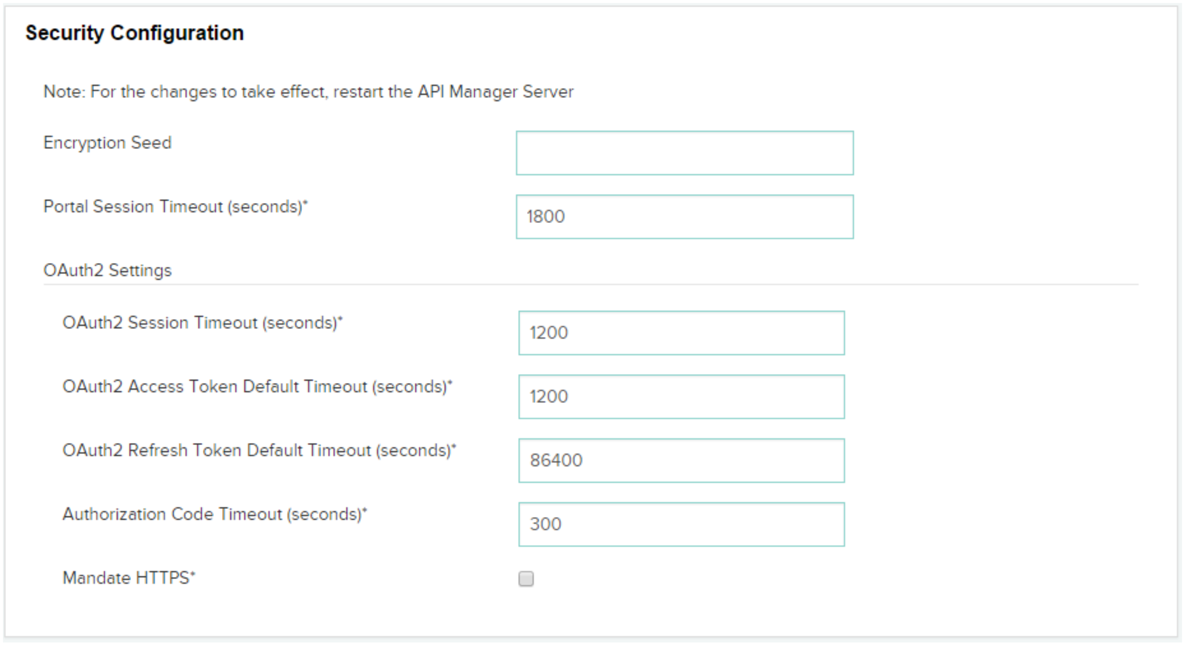
Specifying security settings

Configuration

You can configure the OAuth2 settings on this screen. This protocol allows third-party applications to grant limited access to an HTTP REST service, either on behalf of a resource owner, or by allowing the third-party application to obtain access on its own behalf.

Configuring OAuth2 settings

1 Click Security > Configuration.



Security Configuration

Note: For the changes to take effect, restart the API Manager Server

Encryption Seed	
Portal Session Timeout (seconds)*	1800
OAuth2 Settings	
OAuth2 Session Timeout (seconds)*	1200
OAuth2 Access Token Default Timeout (seconds)*	1200
OAuth2 Refresh Token Default Timeout (seconds)*	86400
Authorization Code Timeout (seconds)*	300
Mandate HTTPS*	☐

2 In the Encryption Seed field, enter a random string as new seed value to encrypt application keys, user store connection passwords, and mail server passwords. This string is always a random value.

***Note:** If you change the encryption seed, restart the API Manager application server.*

3 In the Portal Session Timeout field, enter the number of seconds after which the session of an API publisher or administrator portal application expires.

4 In the OAuth2 Session Timeout field, enter the number of seconds after which your OAuth2 session gets expires. After the session, you will enter your user name and password for obtaining access tokens and authorization codes.

5 In the OAuth2 Access Token Default Timeout field, enter the duration of the default OAuth2 access token after which the original access token expires. A subscriber can override this value in the application configuration.

6 In the OAuth2 Refresh Token Default Timeout field, enter the default OAuth2 refresh token duration after which the ioriginal refresh token expires. A subscriber can override this value in the application configuration.

7 In the Authorization Code Timeout field, enter the number of seconds after which the OAuth2 authorization code expires.

8 To allow only HTTPS on OAuth2 and token endpoints, select the Mandate HTTPS check-box. If you select this check-box, you reject plain HTTP connections.

User Store

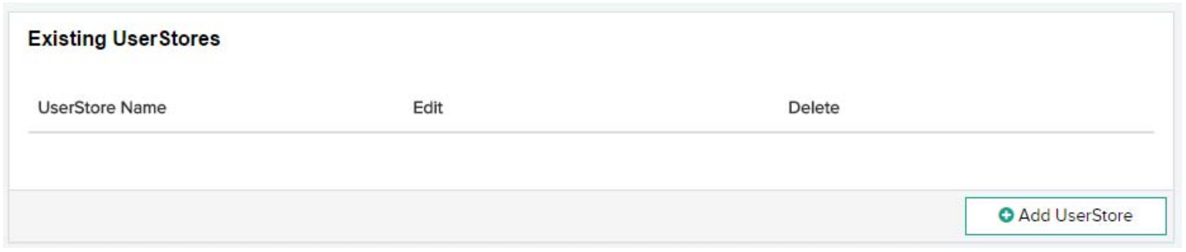
A user store is a database to store information about users and user roles. The database stores login id, password, first name, last name, email address, and so on.

You can create a user store through LDAP or a database connection, such as, JDBC and ODBC. The API Manager comes with an internal user store.

To add a user store:

Creating a user store

1 Click Security > User Store.


Existing UserStores

UserStore Name	Edit	Delete

2 Click Add User Store.

3 From the Connector Type drop-down list, select the type of connector - LDAP Connector or Database Table Connector.

Database Table Connector

You can create a user store using JDBC. You can create a connection string in the database to store the information that an application uses to connect to the database.

Details tab

The Details tab contains the following fields:

Field	Description
Name	The name identifies the user store.

Logon Identifier	The unique identifier for a user store. Prefix this identifier when you log in to a portal.
Description	The information about the user store.
Disabled check-box	Select to enable users in the store access consumer applications.

Connection tab

The Connection tab contains the following fields:

Field	**Description**
Database Server Name	The name of the host from where the database runs.
Database Port	The port number of the database server.
Database User Name	The name of the user that has permission to a table.
Database User Password	The password of the user.
Database Name	The name of the database server that contains the table.
JDBC Driver	The name of the class of the JDBC driver.
JDBC Connection URL	The URL of the JDBC driver. For more information, refer to the JDBC Driver documentation.
Datasource Path	The path to the source of JDBC data. For example, jdbc/SampleDataSource.
Initial JNDI Properties	The list of JNDI standard environment properties.

Configuration tab

The Configuration tab contains the following fields:

Field	**Description**
User Table	The name of the table that contains the user accounts.
Key Column	The value of the column that uniquely identifies the rows in the table.
User First Name Column	The first name of the user.
User Last Name Column	The last name of the user.
User Email Column	The email of the user.
Password Column	The name of the column in the table that holds the values of the passwords of a user.
Roles Table	The name of the table in the database that contain user roles.
Roles Table Key Column	The name of the table that contains the mapping between users and their roles.
User Roles Mapping Table User Key Column	The value of the column that associates user account in the table.
User Roles Mapping Table Role Key Column	The value of the column that associates roles of users in the table.

Pooling tab

The Pooling tab contains the following fields:

Field	Description
Maximum Idle Connections	The maximum number of connections to the database that can be idle.
Minimum Idle Connections	The minimum number of connections to the database that can be idle.
Connection Wait Timeout	The time (in milliseconds) that the pool can wait for a connection before it times out.
Maximum Active Connections	The maximum number of connections that are allocated from the pool concurrently.
Idle Connection Evict Timeout	The time (in milliseconds) to wait before removing an idle connection.

Advanced tab

The Advanced tab contains the following fields:

Field	Description
Enable writing empty string	Select this check-box to write an empty string instead of a null value in columns defined as not-null in the table schema.
Name Quoting	Select this check-box if you want the column names for the database to be within quotes.
Validate Connection Query	The SQL query to validate the database connection.

LDAP Connector

You can configure the API Manager to leverage an LDAP server for user authentication. In this case, the LDAP server creates and manages users.

Details tab

The Details tab contains the following fields:

Field	Description
Name	The name of the user store. The name identifies this user store.
Logon Identifier	The identifier of the user store in LDAP.
Description	The information about this user store.
Disabled	Select this check-box to enable users in the store access the consumer applications.

Connection tab

The Connection tab contains the following fields:

Field	Description
Host	The name or IP address of the LDAP server.
TCP Port	The port number of the LDAP server. This port is the same as the port on which the LDAP listens for SSL connections.

User Bind DN	The bind Distinguished Name (DN) to connect to the LDAP server. The bind DN is the user on the external LDAP server permitted to search the LDAP directory within the defined search base. The role of the bind DN is to query the directory using the LDAP query filter and search for a user. For example, some possible bind DNs are cn=administrator, cn=Users, dc=domain, or c=com.
User Bind DN Password	The password to connect to the LDAP server.
SSL/TLS Enabled	Select this check-box to use a secure connection.
StartTLS Enabled	Select this check-box to allow an application to send secure requests to an LDAP server.
Base Contexts	The points in the LDAP tree for searching the tree.
Failover Servers	The name of all the servers that act as failover servers. For example, "ldap://ldap.example.com:389/" represents a list of failover servers. If the primary server fails, Java Naming and Directory Interface (JNDI) connects to the next available server in the list.

Configuration tab

The Configuration tab contains the following fields:

Field	**Description**
User Configuration	The object classes for creating user objects in LDAP. An object class represents the type of data in an LDAP. An objects class contains attributes of an entry in an LDAP. An objectclass is defined in a schema.
LDAP Filter for Retrieving Accounts	The filter attribute to retrieve user accounts from LDAP.
Account User Name Attributes	The attribute or attributes that represent a user name of an account. An attribute authenticates a user name in an LDAP entry.
User First Name Attribute	The attribute that represents the first name of the user in an LDAP account.
User Last Name Attribute	The attribute that represents the last name of the user in an LDAP account.
User Email Attribute	The attribute that represents the email of the user in an LDAP account.
Group Configuration	The object classes for creating groups in LDAP.
LDAP Filter for Retrieving Groups	The filter attribute to retrieve user groups from LDAP.
Group Name Attribute	The attribute that represents the name of the group in an LDAP account.
Group Membership Attribute	The attribute that contains users in an LDAP group.

Advanced tab

The Advanced tab contains the following fields:

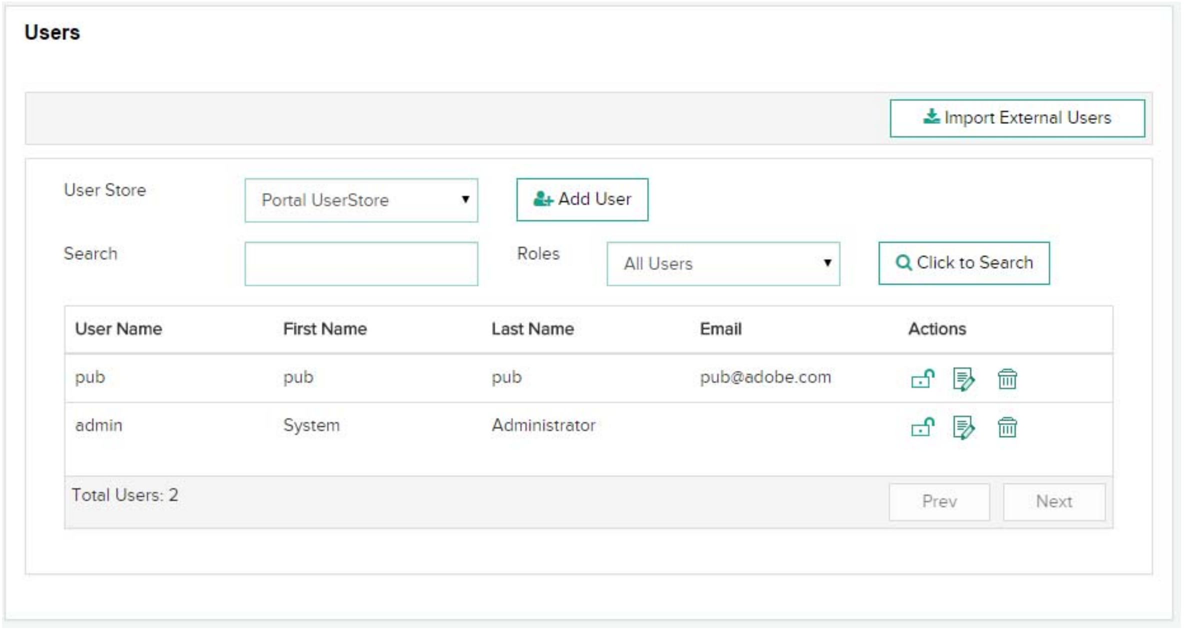
Field	Description
Use Paged Result Control	Select this check-box to ensure that the LDAP uses paged results instead of Virtual List View (VLV) when retrieving user accounts. VLV lets you query a large directory in chunks. For example, let there be a directory with a large number of users, "ou": dc=demo,dc=local + ou=DemoGroups (100,000 groups) + ou=DemoUsers (100,000 users) If you want to present the information in the user group "ou" in a scrollable or paged window in an application, retrieving 1000 pages of 100 results each is an inefficient way. This method is resource-intensive and wastes bandwidth. On the other hand, VLV queries a large data-set with a sorting rule. For example, using VLV, you can sort the first 50 users ordered by organizationName in an LDAP.
LDAP Referral Handling	Either **follow** or **ignore** LDAP referrals. LDAP referrals enable an LDAP tree to be distributed across multiple LDAP servers. Therefore, an LDAP server can reference other LDAP servers even though it does not store the full Directory Information Tree (DIT). When you browse a particular directory, an LDAP server returns referrals after referring you to another server in the tree.

Adding users

User management in the API Manager involves defining and managing users, roles, and their access levels. You can add or delete users, assign roles, search for users, manage user stores, and import external users from LDAP or database connection.

To add a user:

1 Click Security > Users

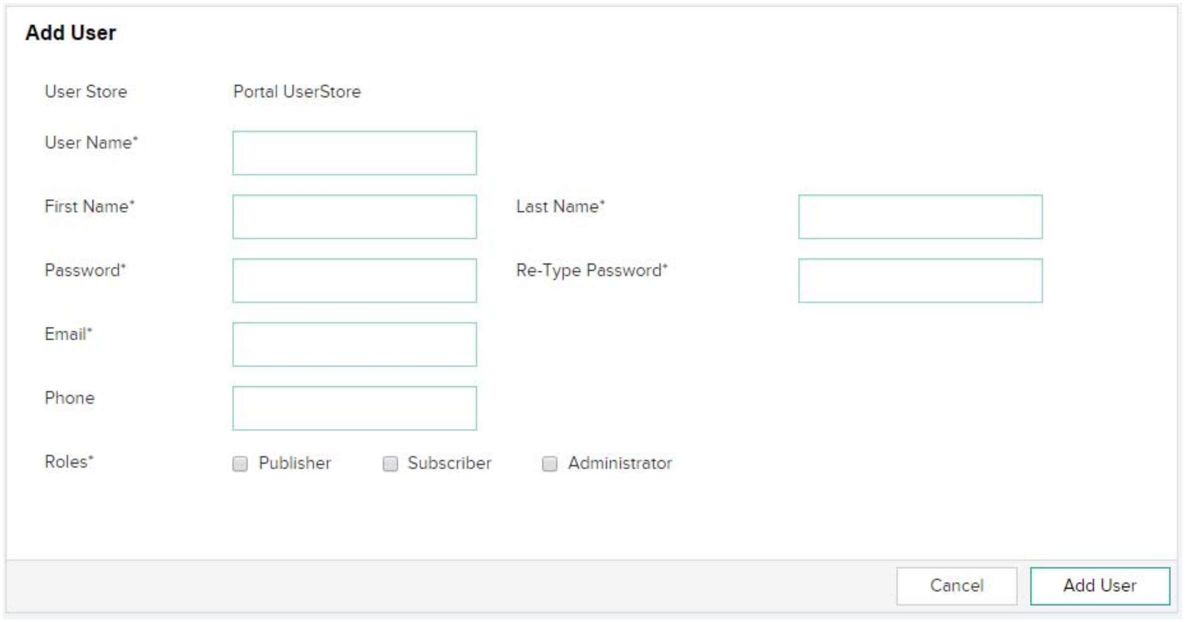


User Name	First Name	Last Name	Email	Actions
pub	pub	pub	pub@adobe.com	
admin	System	Administrator		

Total Users: 2

2 Click Import External Users to retrieve users from a database or an LDAP connection.

3 To add a user, select a user store and click **Add User**. The following page displays.



a In the **User Name** field, enter the unique identifier of the new user.

b Enter the attributes of the user in the rest of the fields.

c Select the appropriate role for the new user- Publisher, Subscriber, or Administrator. For more details, see Roles .

d Click **Add User** to add the user in the user store.

Configuring SLAs

In the API Manager, an application or a resource can be available to a consumer at different levels of service. SLA configuration enforces access control through rate limiting and throttling. Throttling tiers limit the number of hits to an API over a certain time, due to monetization of the API, security restrictions, infrastructure issues, and so on.

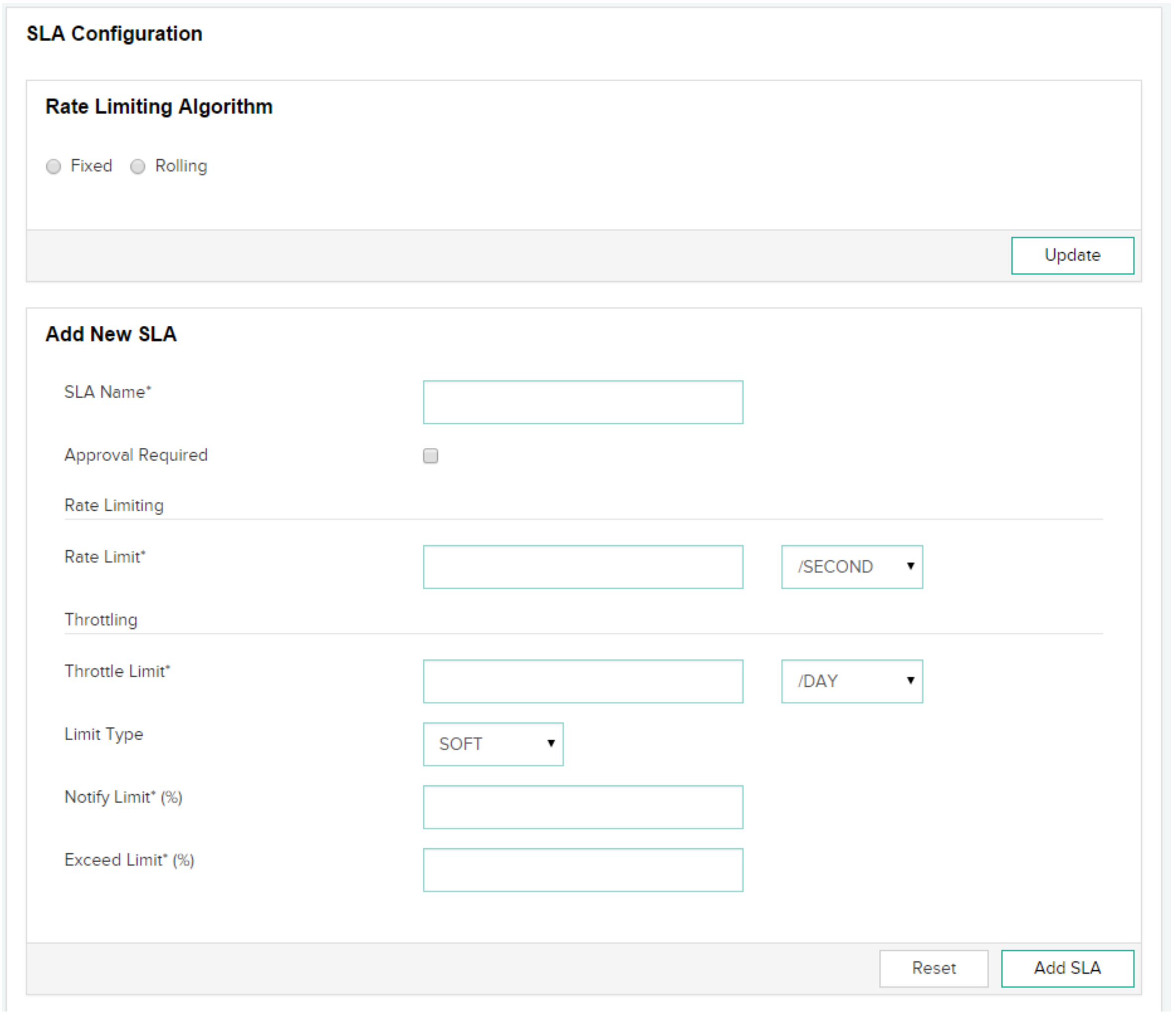
For example, you can set throttling tiers to the APIs to limit access to it accordingly. Each tier defines a maximum number of requests per day or week or month.

You can also set the rate limit for an SLA per second or minute or hour.

You can then add your own tiers to the API Manager using the instructions below:

Adding an SLA

1 Click SLA Configuration. The following page displays.


SLA Configuration

Rate Limiting Algorithm

Fixed Rolling

Update

Add New SLA

SLA Name*		
Approval Required	☐	
Rate Limiting		
Rate Limit*		/SECOND
Throttling		
Throttle Limit*		/DAY
Limit Type	SOFT	
Notify Limit* (%)		
Exceed Limit* (%)		

2 Select the rate limiting algorithm. In API manager, two types of algorithms can be used to limit the rate: Rolling and Fixed.

In **fixed window** algorithm, the period is considered from the starting of the time unit to the end of the time unit. For example, a period is considered as 0-60 seconds for a minute irrespective of the time frame at which the API request has been made.

In **rolling window** algorithm, the period is considered from the fraction of the time at which the request has been made to the end of the time unit. For example, if two requests for API calls are made at 30th second and 40th second of a minute it is considered as two requests from 30th second of that minute up to the 30th second of next minute.

For more details, refer to Throttling and Rate-limiting.

3 In the **SLA Name** field, enter the name of the new SLA.

In the **Rate Limit** field, enter the API request rate limit in the SLA. You can only limit an API request rate to number of requests per second, minute, or hour.

In the **Throttle Limit** field, enter the API throttle limit in the SLA. You can only limit an API request rate to number of requests per second, minute, or hour.

Select the type of throttle limit- Hard or Soft.

- **Hard:** If you select this option, the number of API requests cannot exceed the throttle limit.
- **Soft:** If you select this option, you can set the API request limit exceed a percentage. For example, if you set the Exceed Limit to 90%, the number of API requests can exceed 90% of the prescribed limit.

4 In the **Notify Limit** field, enter the limits (in percentage) at which you receive notifications when the number of API requests approach the limit.

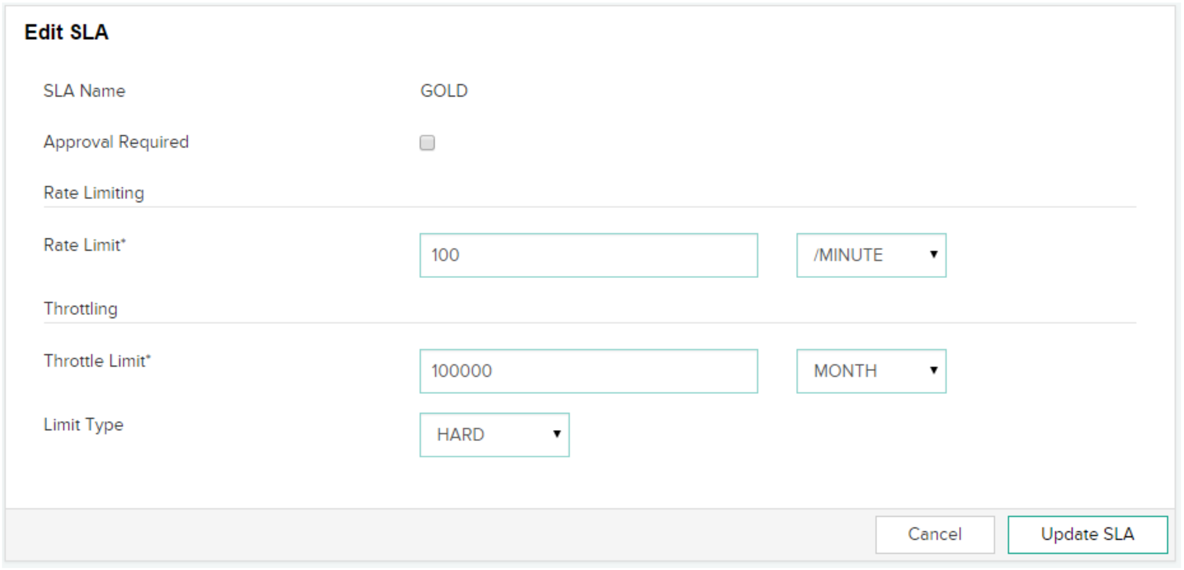
5 In the **Exceed Limit** field, enter the limits (in percentage) by which the number of API requests can exceed.

6 If the SLA requires an approval from the API publisher, select the **Approval Required** check-box.

Editing an SLA

You can edit an SLA and modify its properties. Choose an SLA from the Existing SLAs list.

1 Click **Edit**. The following page displays.



Edit SLA

SLA Name: GOLD

Approval Required: ☐

Rate Limiting

Rate Limit*: 100 /MINUTE

Throttling

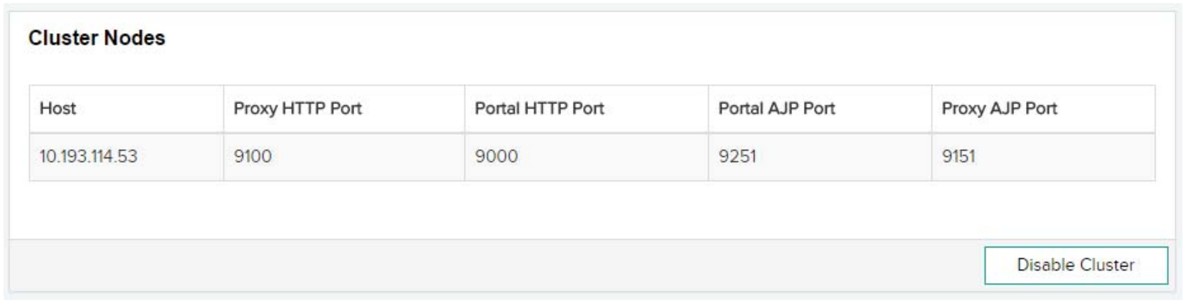
Throttle Limit*: 100000 MONTH

Limit Type: HARD

2 Update all the properties of the SLA and click **Update SLA**.

Viewing cluster information

You can see a list of all clusters configured in API Manager. You can also see a cluster's proxy port, portal port, and so on.

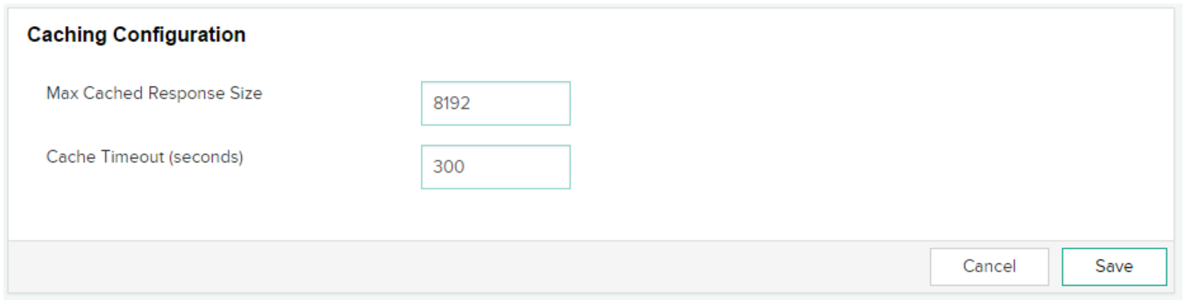


Cluster Nodes

Host	Proxy HTTP Port	Portal HTTP Port	Portal AJP Port	Proxy AJP Port
10.193.114.53	9100	9000	9251	9151

Caching

Set the maximum response size to define the maximum size, in bytes, of the response that can be stored in the output cache. Also specify the timeout, in seconds.

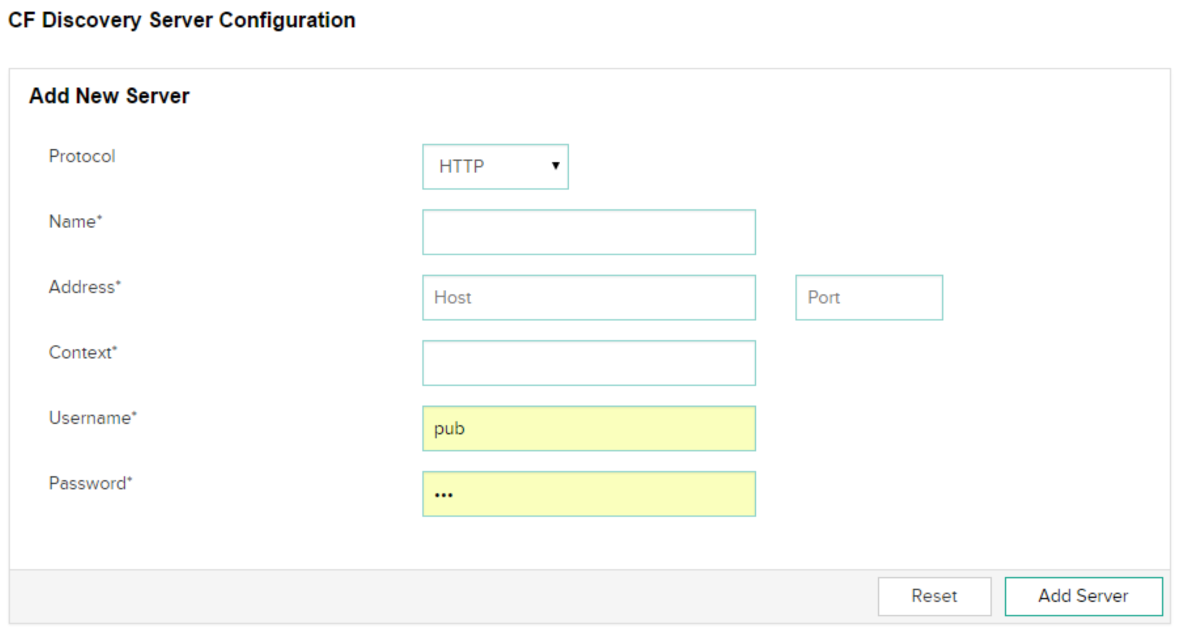


Caching Configuration

Max Cached Response Size	8192
Cache Timeout (seconds)	300

ColdFusion discovery server

If you want to discover REST services published in a ColdFusion non-InVM environment, configure the ColdFusion server on the **CF Discovery Server Configuration** page.



CF Discovery Server Configuration

Add New Server

Protocol	HTTP
Name*	
Address*	Host / Port
Context*	
Username*	pub
Password*	•••

1 In the **Protocol** drop-down list, select HTTP or HTTPS.

2 In the **Name** field, enter the name of the ColdFusion server.

3 In the **Address** field, enter the host and port number of the CF server.

4 In the **Context** field, enter the REST servlet context of the CF server.

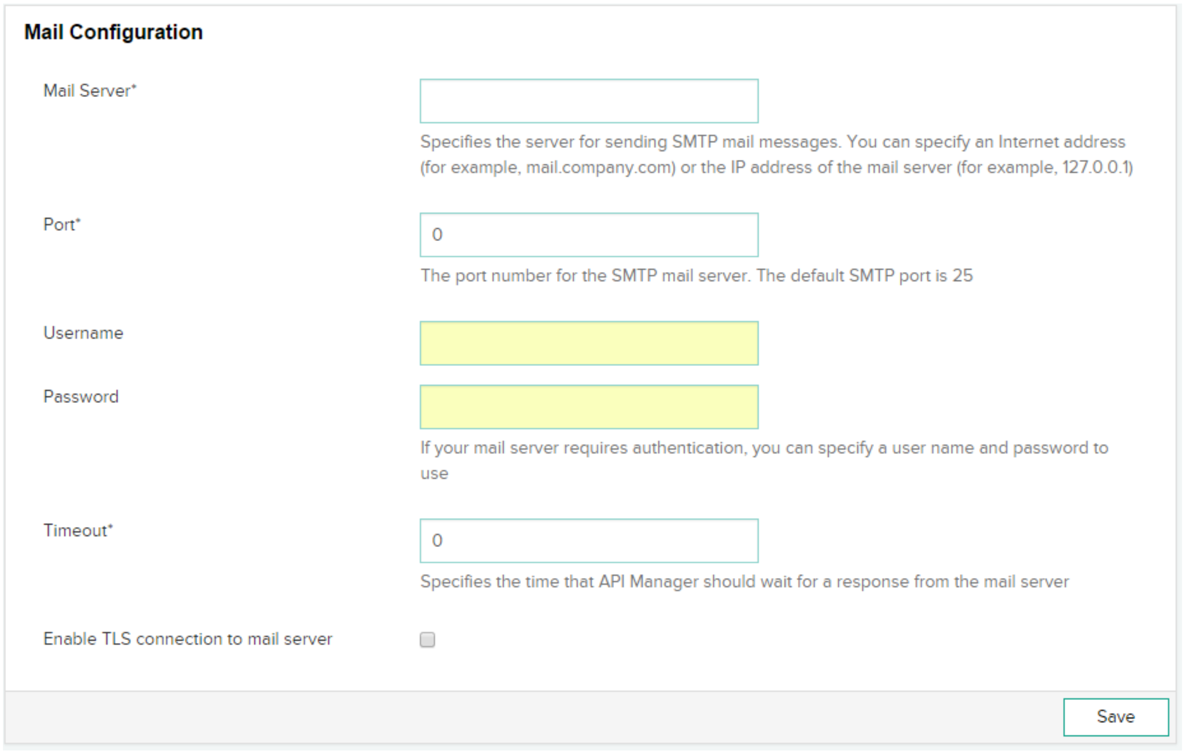
5 In the **Username** and **Password** fields, enter the credentials to log in to the ColdFusion server.

Configuring notifications

The API Manager sends notifications for any alerts or user events that require attention. You can configure the settings for email notifications for publishers and consumers.

Configuring mail settings

1 Click Notifications > Mail. The following page displays.



Mail Configuration

Mail Server*

Specifies the server for sending SMTP mail messages. You can specify an Internet address (for example, mail.company.com) or the IP address of the mail server (for example, 127.0.0.1)

Port* 0

The port number for the SMTP mail server. The default SMTP port is 25

Username

Password

If your mail server requires authentication, you can specify a user name and password to use

Timeout* 0

Specifies the time that API Manager should wait for a response from the mail server

Enable TLS connection to mail server

2 In the Mail Server field, enter the name of the server that sends Simple Mail Transfer Protocol (SMTP) messages. Alternatively, you can enter the mail server (for example, mail.company.com) or the IP address of the mail server (for example, 127.0.0.1).

3 In the Port field, enter the port number of the mail server.

4 In the Username and Password fields, enter the credentials to authenticate a user in the mail server.

5 In the Timeout field, enter the time (in seconds) that the API Manager waits for a response from th email server.

6 In the Protocol field, enter the messaging protocol for the mail server, for example, SMTP.

7 To enable Transport Level Security (TSL) on the connection to the mail server, select the Enable TLS connection to mail server check-box. Transport Layer Security (TLS) is a protocol that ensures privacy between applications and their users within a network. When a server and client communicate, TLS ensures security of the communication.

Configure log

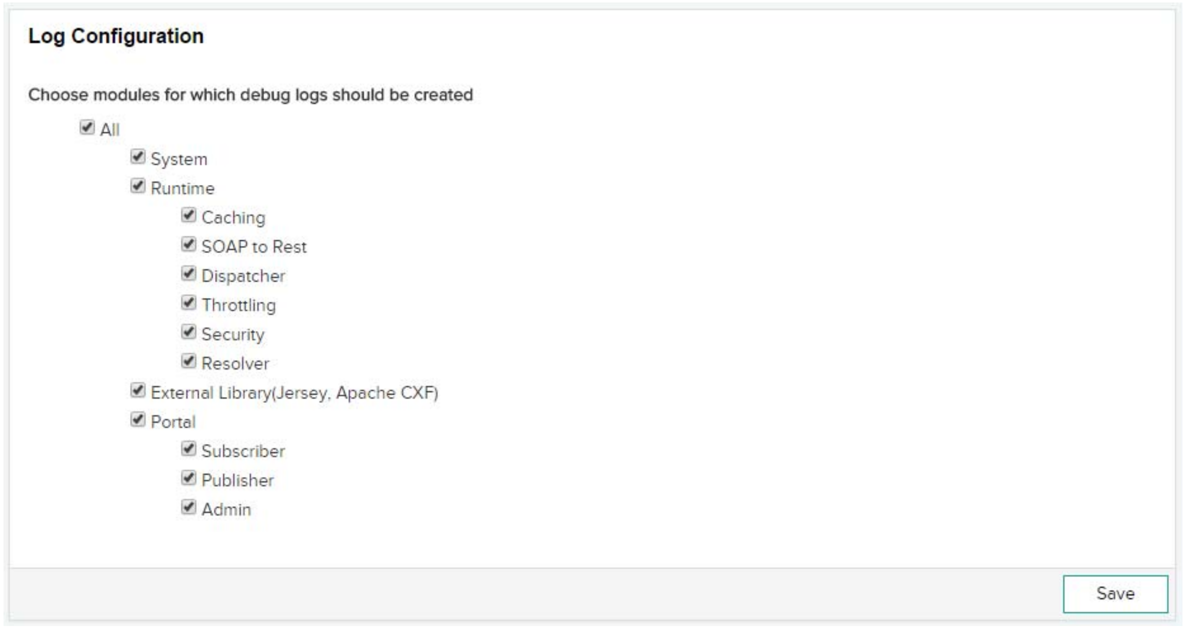
In this page, choose the modules for which you want to generate debug logs. For example,

- Messages logged during startup, shutdown, and cluster updates.

- Messages logged when a request is dispatched.
- Messages logged when an API resolves a request.

You can enable debugging for specific sections that you want to debug. For example, if you want to check for any issues while creating an API, enable the **Publisher** check-box in the Portal section.

This way you can manage your debug logs properly.



Log Configuration

Choose modules for which debug logs should be created

- [✓] All
  - [✓] System
  - [✓] Runtime
    - [✓] Caching
    - [✓] SOAP to Rest
    - [✓] Dispatcher
    - [✓] Throttling
    - [✓] Security
    - [✓] Resolver
  - [✓] External Library(Jersey, Apache CXF)
  - [✓] Portal
    - [✓] Subscriber
    - [✓] Publisher
    - [✓] Admin

Viewing the analytics dashboard

Analytics dashboard helps you view the number of APIs that are run from a node, the number of API requests, and system statistics. The API Manager collects information as APIs process information. If you see a sudden spike on a graph or chart, you can take immediate measures.

By clicking a particular data (for example, API name or Publisher name or field in graph), you can filter the entire dashboard on that field.

Click Analytics to display the dashboard.

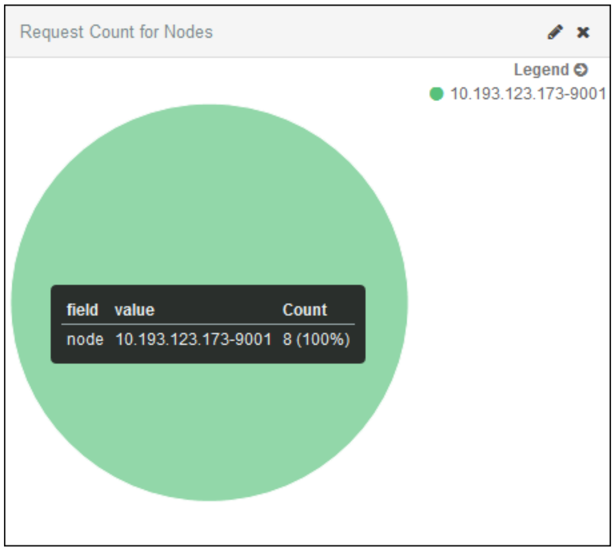
There are three types of dashboards in the Administrator portal:

- Home
- Publisher APIs
- Publishers

Home: On the Home dashboard panel, you can see the following dashboards:

Request Count for Nodes

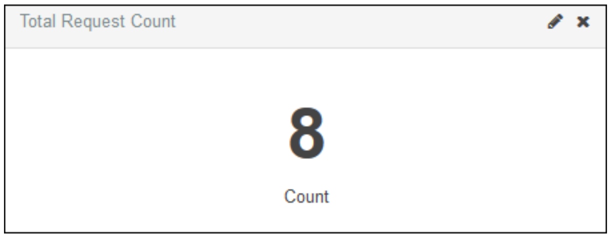
You can view the node IP and port on which an API runs. You can also view the number of requests by the API. Hover the pointer on the visualization to see the details.



field	value	Count
node	10.193.123.173-9001	8 (100%)

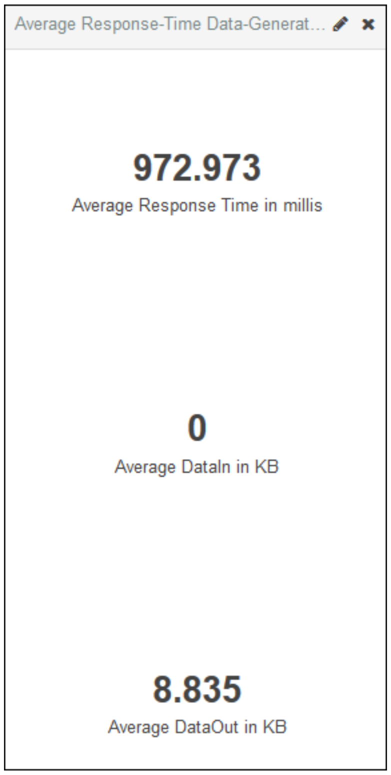
Total Request Count

You can view the count of requests by an API.



Consumed Metrics

You can view the avarage response time (in milliseconds) of an API, and the average incoming and outgoing data (in KB) of the API.


Average Response-Time Data-Generat...

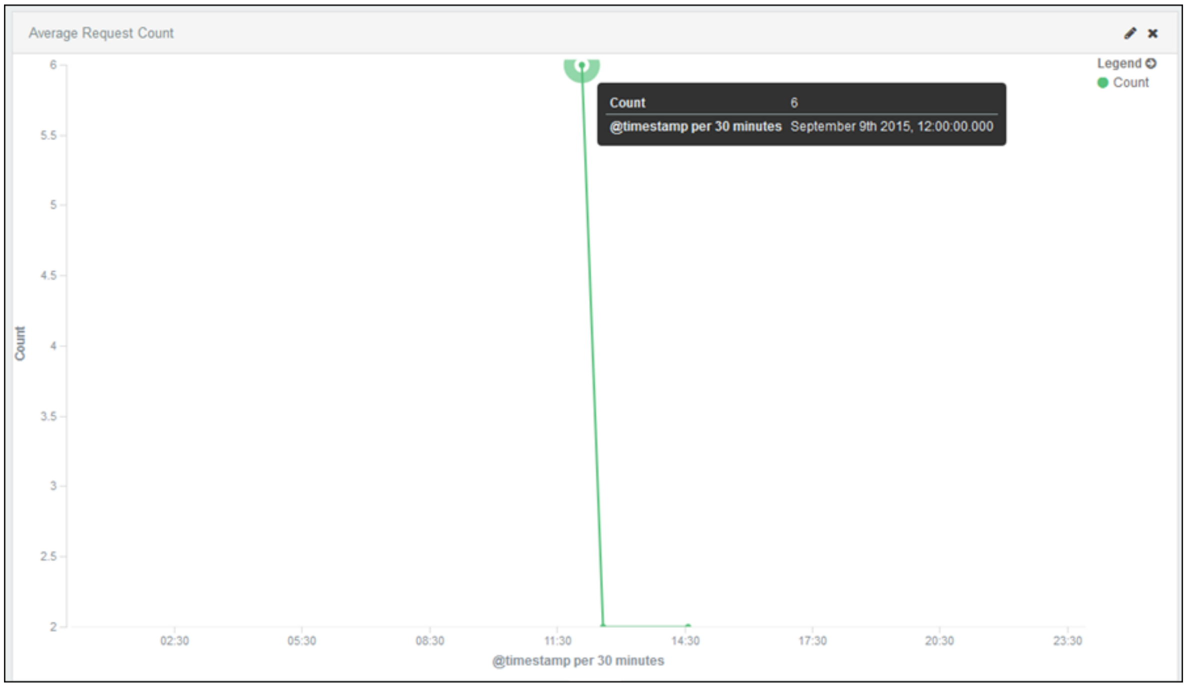
972.973
Average Response Time in millis

0
Average DataIn in KB

8.835
Average DataOut in KB

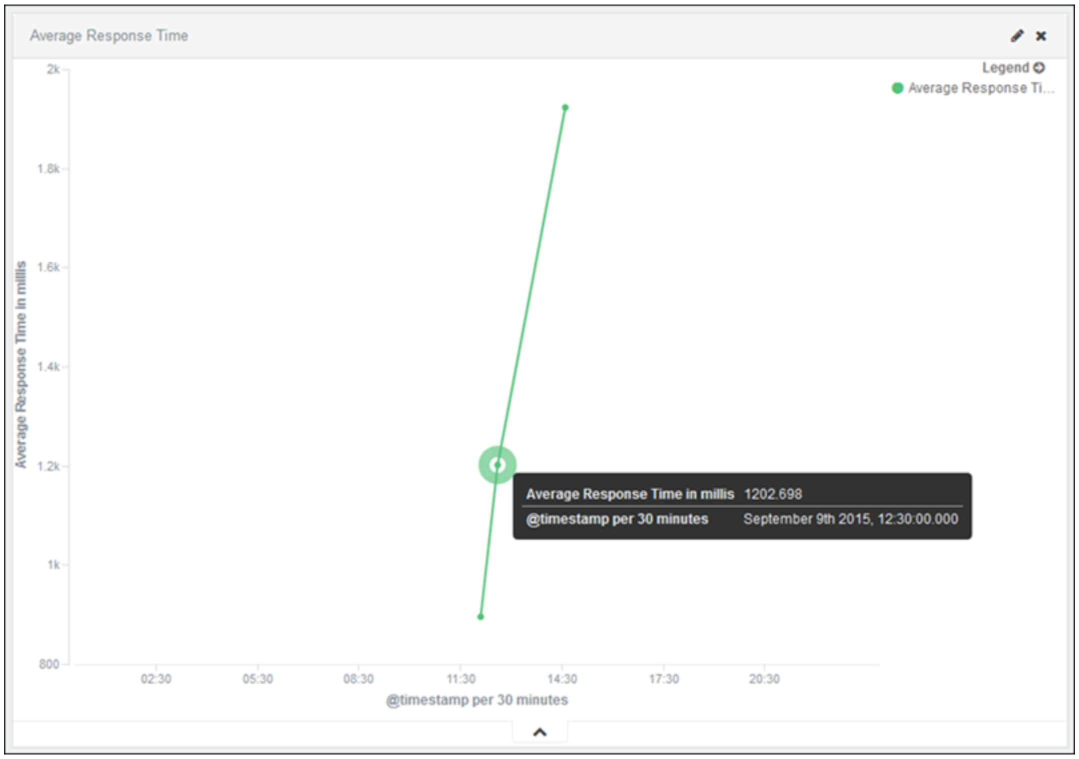
Average Request Count

You can view a line graph of the count of API requests. You can further filter the data according to the time range, for example, today, this week, this month, last minutes, last 30 minutes, last 5 years, and so on. Hover the pointer on an point on the axis to view the number of API requests within the time range.



Average Response Time

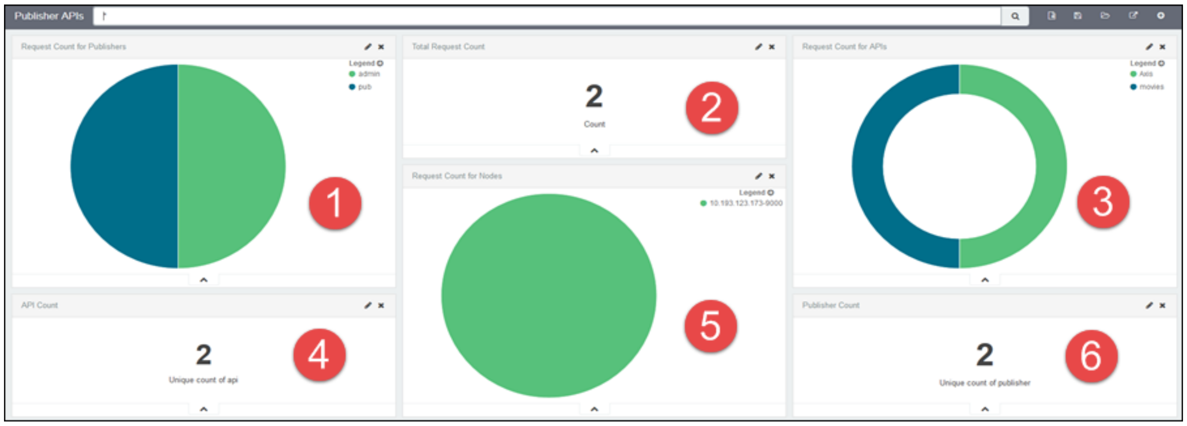
You can view the average response time of an API using this visualization. This time is in milliseconds. You can filter the data according to a time range.



Publisher APIs: On the Publisher APIs dashboard, you can see the details of each API from a publisher, the number of APIs, the number of publishers, the response times for APIs, and so on.

You can filter the requests and whole dashboard on the publishers.

Hover the pointer on a pie chart or a line chart to view more details.



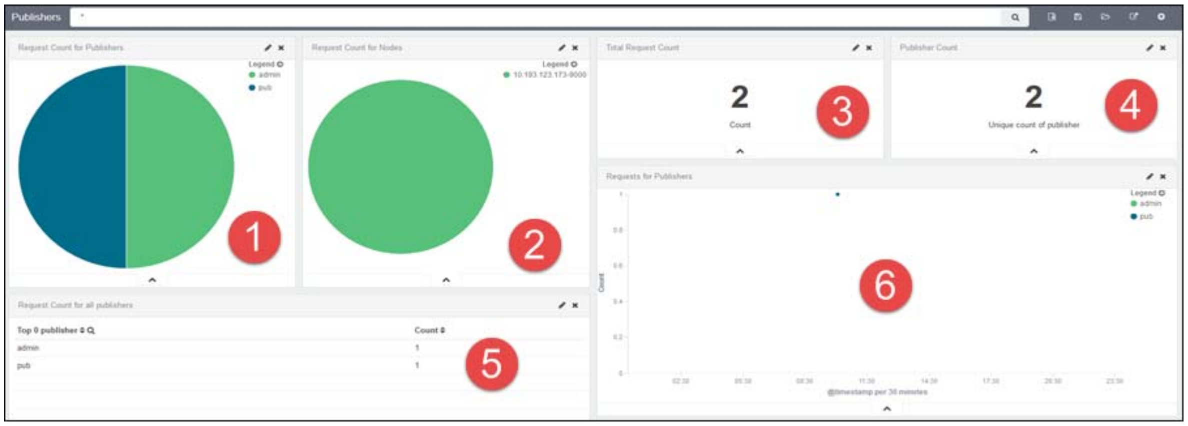
1	Pie-chart for the number of API requests by publishers.
2	The number of API requests by a publisher.
3	The number of API requests.

4	The number of APIs.
5	The number of nodes in a cluster that contains the APIs.
6	The number of publishers.

Publishers: On the Publishers dashboard, you can see the visualization of the number of API requests, the number of publishers, average data consumption per publisher, and so on.

You can filter the requests and whole dashboard on the publishers.

Hover the pointer on a pie chart or a line chart to view more details.

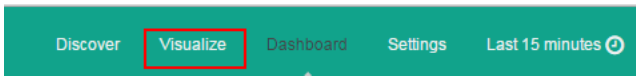


1	Pie-chart for the number of API requests by publishers.
2	The number of nodes in a cluster that contains the APIs.
3	The number of API requests.
4	The unique count of publishers that have made at least one request within the time range.
5	The number of API requests from all publishers.
6	The number of API requests by publishers according to time-range.

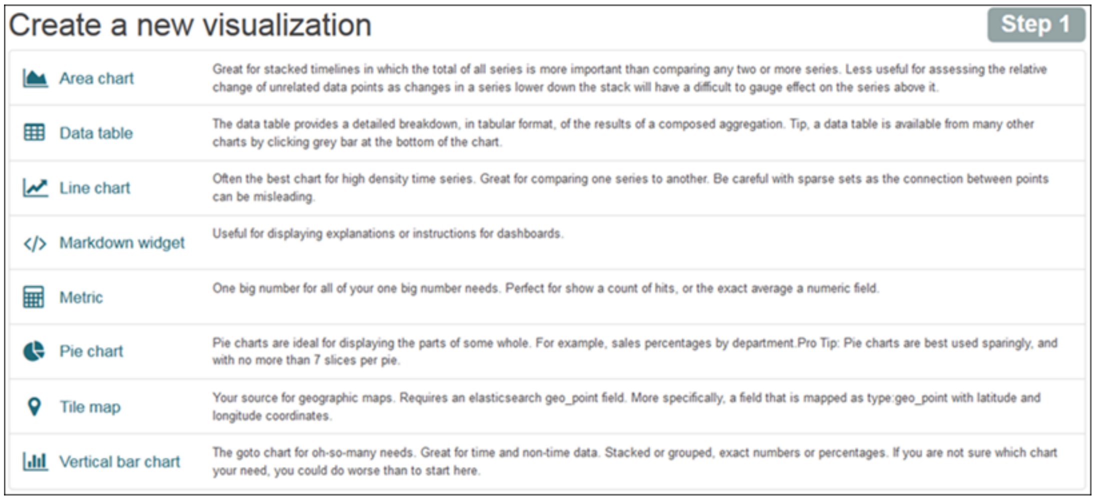
Creating a visualization

You can create your custom visualizations and save it for further analysis. Follow the steps below to create a visualization:

1 To create a visualization, click the **Visualize** tab.



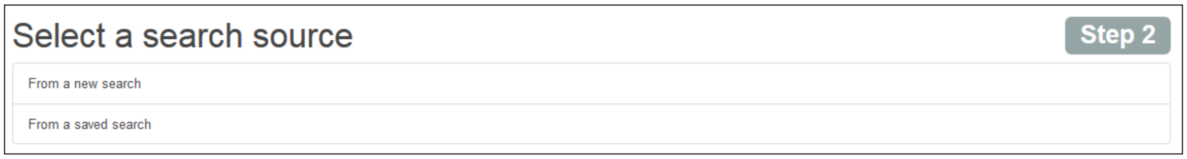
2 You can create visualizations in the form of area charts, data table, pie charts, and so on, from API information. As an example, create an area chart from the API analytics. Click Area chart from the list.



Create a new visualization Step 1

Area chart	Great for stacked timelines in which the total of all series is more important than comparing any two or more series. Less useful for assessing the relative change of unrelated data points as changes in a series lower down the stack will have a difficult to gauge effect on the series above it.
Data table	The data table provides a detailed breakdown, in tabular format, of the results of a composed aggregation. Tip, a data table is available from many other charts by clicking grey bar at the bottom of the chart.
Line chart	Often the best chart for high density time series. Great for comparing one series to another. Be careful with sparse sets as the connection between points can be misleading.
Markdown widget	Useful for displaying explanations or instructions for dashboards.
Metric	One big number for all of your one big number needs. Perfect for show a count of hits, or the exact average a numeric field.
Pie chart	Pie charts are ideal for displaying the parts of some whole. For example, sales percentages by department.Pro Tip: Pie charts are best used sparingly, and with no more than 7 slices per pie.
Tile map	Your source for geographic maps. Requires an elasticsearch geo_point field. More specifically, a field that is mapped as type:geo_point with latitude and longitude coordinates.
Vertical bar chart	The goto chart for oh-so-many needs. Great for time and non-time data. Stacked or grouped, exact numbers or percentages. If you are not sure which chart your need, you could do worse than to start here.

3 Select from the following:

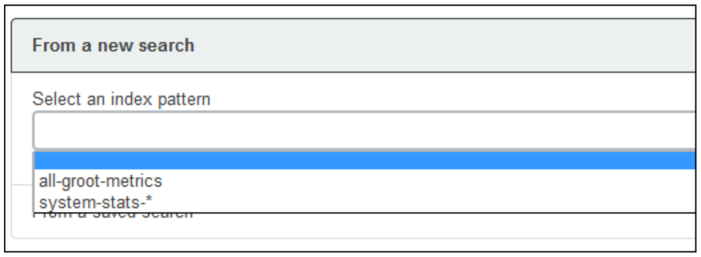


Select a search source Step 2

From a new search

From a saved search

4 Create a visualization from a new search. Select From a new search. You can see the following index patterns in the drop-down list:



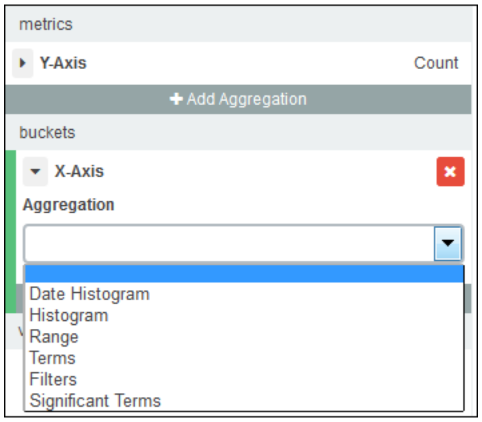
From a new search

Select an index pattern

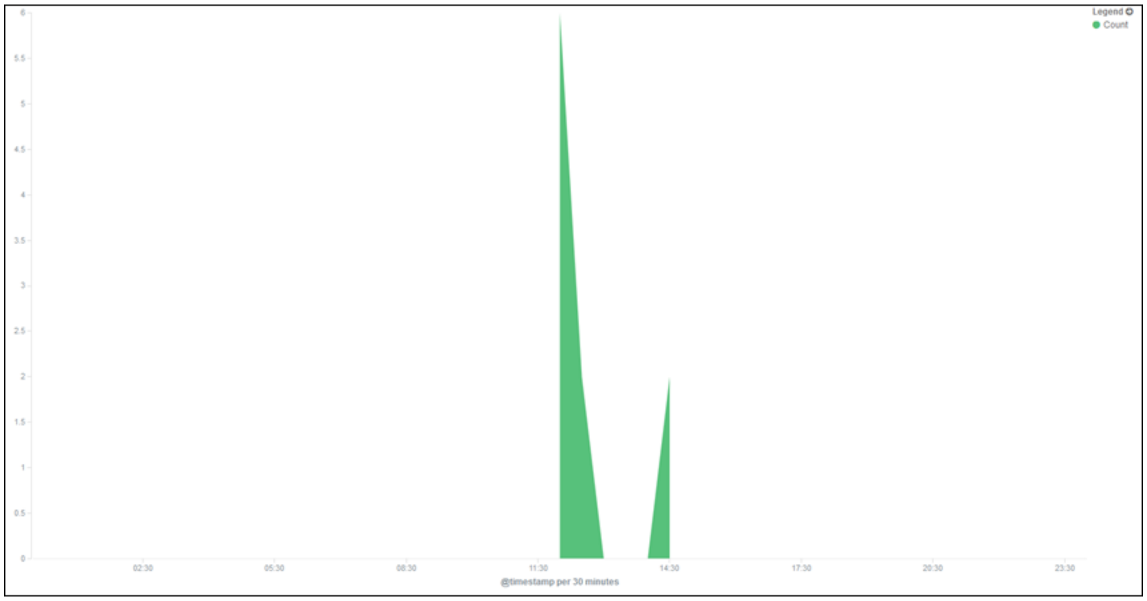
all-groot-metrics
system-stats-*

From a saved search

5 Select the data point on the X-Axis and click Apply.



An area chart displays with number of API requests on the Y-axis and API time-stamp on the X-axis.



Filtering data according to time range

In the Analytics page, you can filter the results according to a time range.

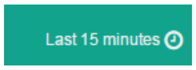
There are three ways options:

1 Quick

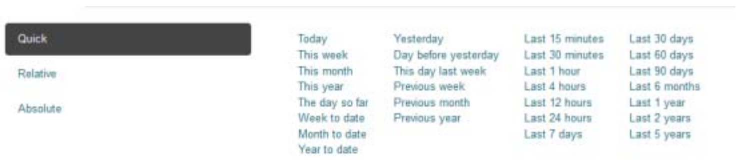
2 Relative

3 Absolute

1 Click the time filter as shown below:

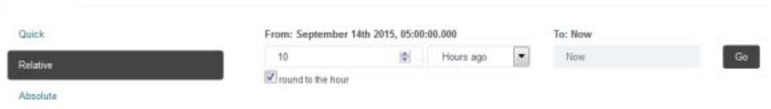


2 Select any time range from the list, as shown below:



If you select **Today**, you can see the analytics of all APIs for the last 24 hours.

3 Click **Relative**. You can filter the data from a specified date and time to the current date and time.



From: September 14th 2015, 05:00:00.000

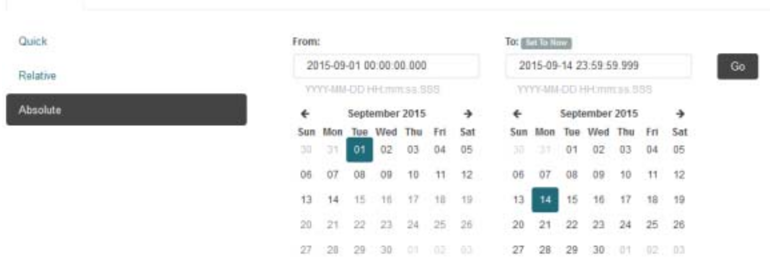
10 Hours ago

☑ round to the hour

To: Now

Now

4 Click **Absolute**. You can filter the results according to dates.



From: 2015-09-01 00:00:00.000

YYYY-MM-DD HH:mm:ss.SSS

To: 2015-09-14 23:59:59.999

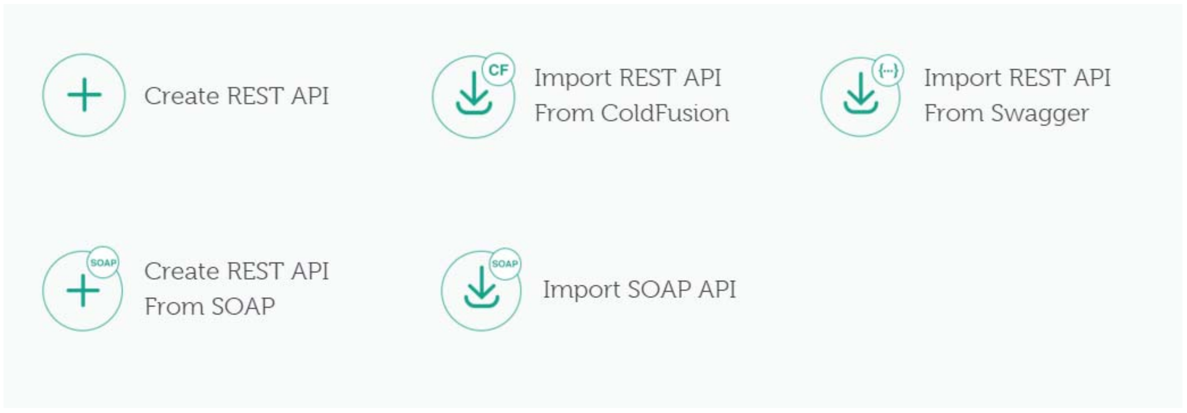
YYYY-MM-DD HH:mm:ss.SSS

Publisher

Overview

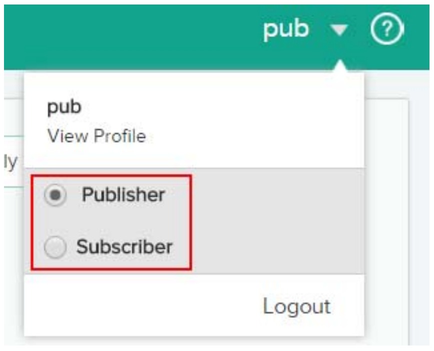
As a publisher, you can create and publish an API for consumption by any service or application. Subscribers can consume the APIs and use the APIs in their applications. Using the portal, you can design APIs and manage the APIs throughout their lifecycle interactively. You can collaborate with consumers, engage developers, and gather important information and insights necessary to get your API up and running.

When you log in to the API Manager portal as a publisher, you can see the following options:


Create REST API
Import REST API From ColdFusion
Import REST API From Swagger
Create REST API From SOAP
Import SOAP API

Switching between roles

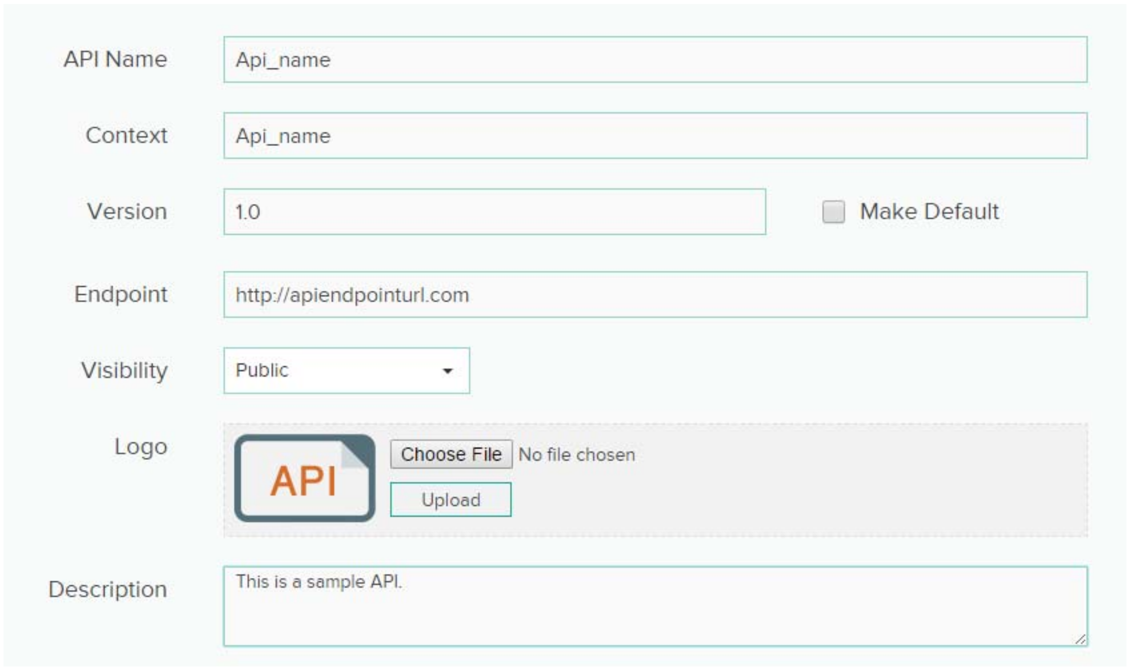
When the API Manager administrator assigns more than one role (for example, publisher and subscriber) to you, you can switch between the roles and log in as publisher or subscriber.



Creating a REST API

Follow these steps to create a REST API in API Manager.

1 Click **Create REST API**. The Basic Settings page displays.



API Name: Api_name

Context: Api_name

Version: 1.0 Make Default

Endpoint: http://apiendpointurl.com

Visibility: Public

Logo: No file chosen

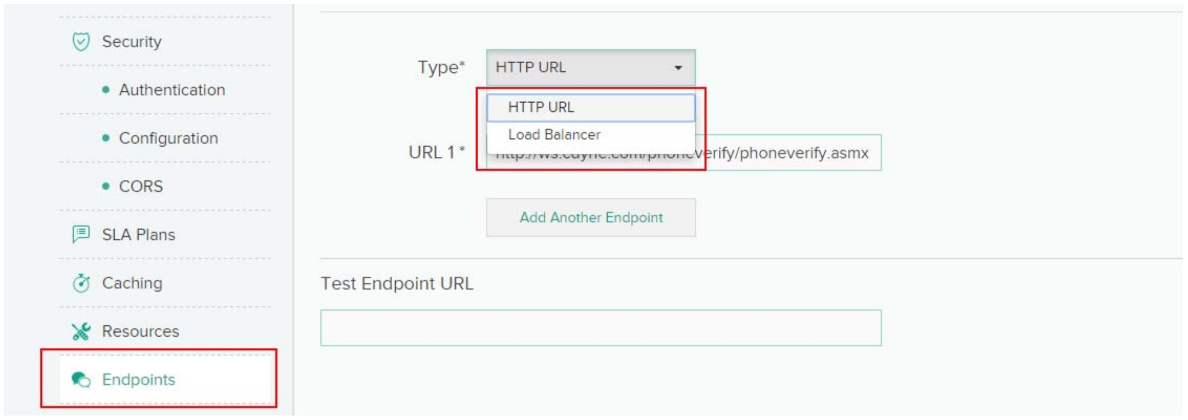
Description: This is a sample API.

2 Enter the following details in the Basic Settings section.

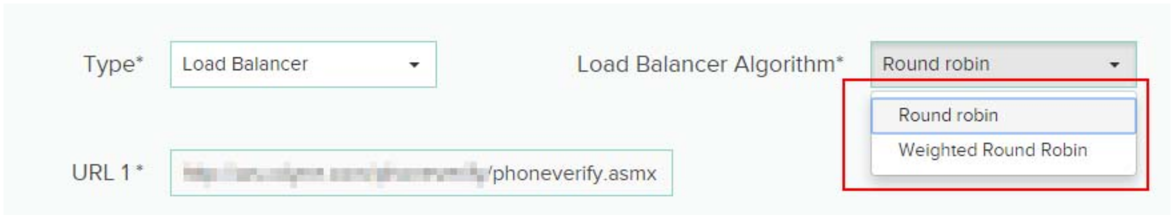
Field	Description
API Name	Name of the new API. Once you create an API, you cannot change its name.
Context	Defines a context for a service request. Resources are viewed hierarchically via their URI names, offering consumers a friendly, easily understood hierarchy of resources to leverage in their applications.
Description	The description of the API.
Visibility	Visibility settings prevent certain roles from viewing and modifying APIs from another user. Select Public, Intranet, or Partner modes of publishing an API. In Public mode, all levels of users can see an API.
Version	The version of the API. An API can have multiple versions. The version must always begin with v or V followed by a whole number or decimal. The version can also begin without v or V. For example,
Make Default	Select this check-box if you want your API version to be the default. If you have multiple versions of the same API, you can choose the one to be the default.
Lifecycle	Stages of progression of an API. Select from Draft, Published, Deprecated, or Retired.

3 Define the API endpoints. An endpoint defines the address to a web service. You can expose both REST and SOAP services to consumers through APIs. Enter the following details in the Endpoints section.

4 To choose the type of endpoint- HTTP URL or Load Balancer, click **Endpoints**.

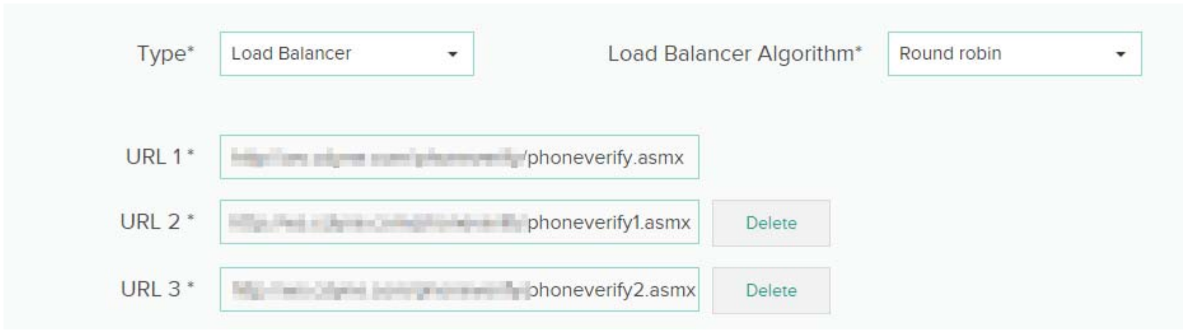


5 If you select **Load Balancer**, choose the load balancing algorithm.

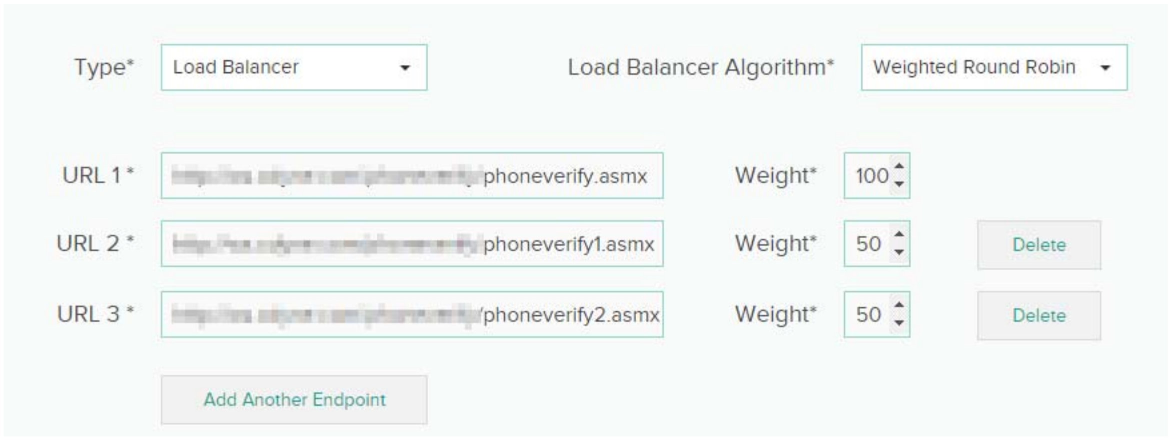


There are two load balancing algorithms that the API Manager uses:

- **Round robin:** In this method, the load is balanced sequentially across the API endpoints. All the endpoints are assigned similar load. The endpoints also similar performance. Click **Add Another Endpoint** to add the endpoint URLs for load distribution.



- **Weighted Round Robin:** In this method, the load is balanced across the endpoints according to a "weightage" factor that you can assign to each endpoint. The load is divided in terms of the weights assigned. If the weights assigned to the three endpoints are 100, 50, and 50 respectively, the first endpoint handles twice as many requests as the second third endpoints. In the following example:
  - The first endpoint handles the first two requests.
  - The second endpoint handles the third request.
  - The third endpoint handles the fourth request.



Type* Load Balancer

Load Balancer Algorithm* Weighted Round Robin

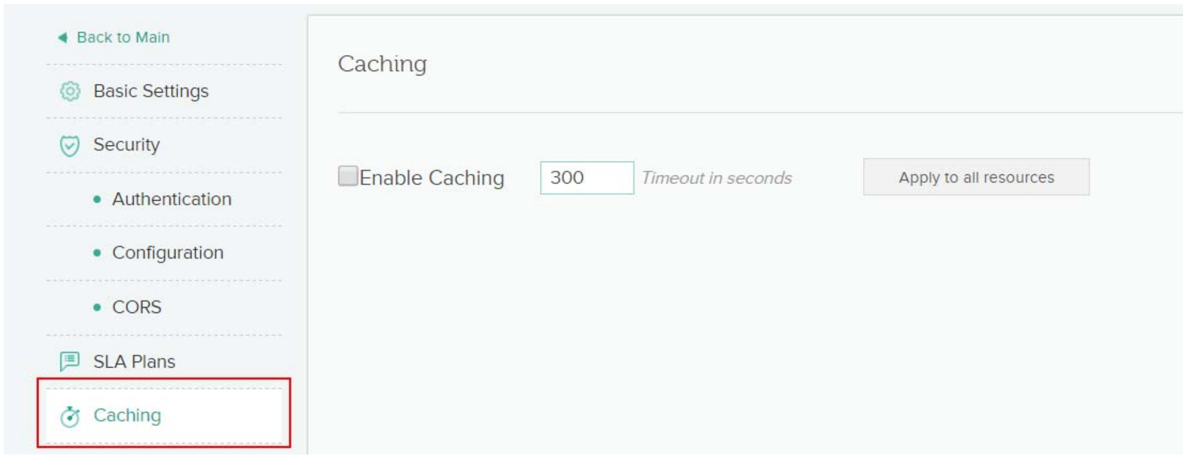
URL 1 * phoneverify.asmx Weight* 100

URL 2 * phoneverify1.asmx Weight* 50

URL 3 * /phoneverify2.asmx Weight* 50

6 To enable caching of your newly created REST API, click Caching. The benefits of caching an API are speed and reduced load on server. Select the **Enable Caching** check-box and enter the caching timeout in seconds.

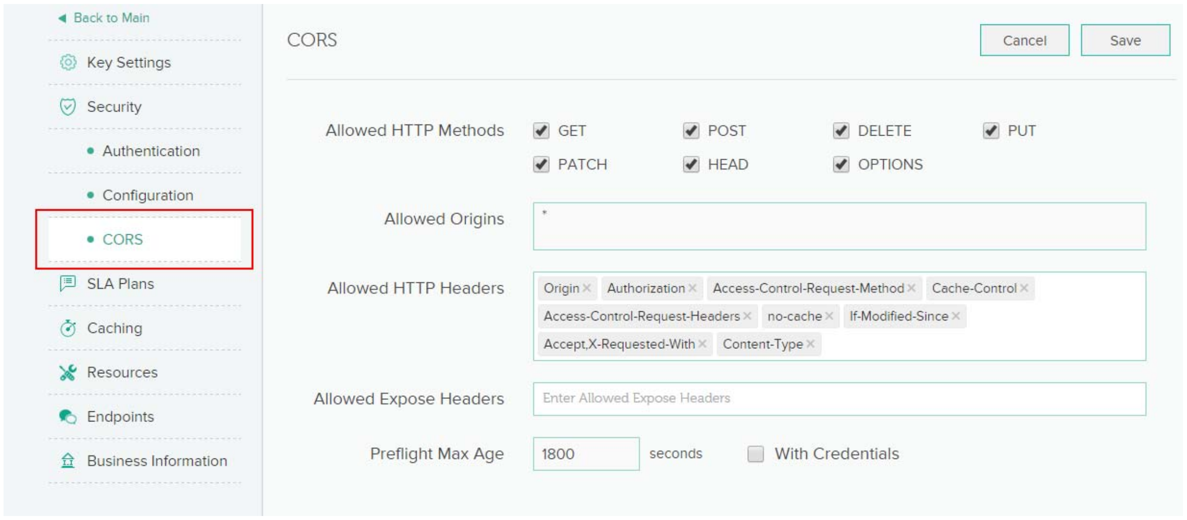
To enable caching at resource level, select the **Apply to all resources** check-box. Caching at resource level prevents a separate back-end call to check the authorization type and throttling level of a resource, every time a request to the API comes.



Enable Caching 300 Timeout in seconds

7 To enable Cross-Origin Resource Sharing (CORS) of the API, click CORS. CORS allows restricted resources in the API to be accessed from another domain. CORS is a standard mechanism that allows JavaScript calls executed in a web page to interact with resources originating from an external domain. All known browsers enforce this policy.

To know more about CORS, refer to CORS .



CORS

Allowed HTTP Methods: [✓ GET] [✓ POST] [✓ DELETE] [✓ PUT] [✓ PATCH] [✓ HEAD] [✓ OPTIONS]

Allowed Origins: *

Allowed HTTP Headers: Origin, Authorization, Access-Control-Request-Method, Cache-Control, Access-Control-Request-Headers, no-cache, If-Modified-Since, Accept,X-Requested-With, Content-Type

Allowed Expose Headers: Enter Allowed Expose Headers

Preflight Max Age: 1800 seconds

With Credentials: [ ]

Adding resources to the API

In the Resources section, you can separate your API into resources. You can access or manipulate these resources using HTTP requests where each method, GET, PUT, or POST, and so forth, performs to its specifications.

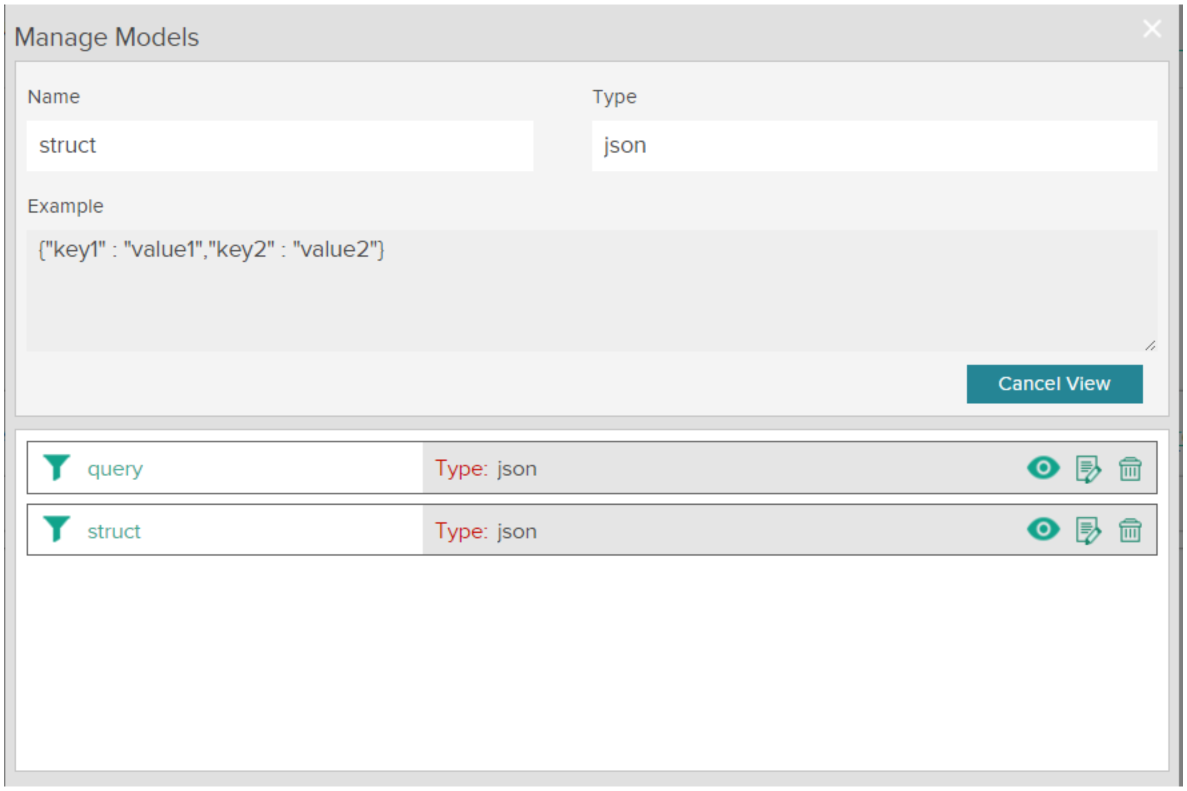
A resource is typically defined as an object with a type and the methods that operate on it.

Resources can be grouped into collections. There are also singleton resources that exist outside a collection.

To add a resource to the new API:

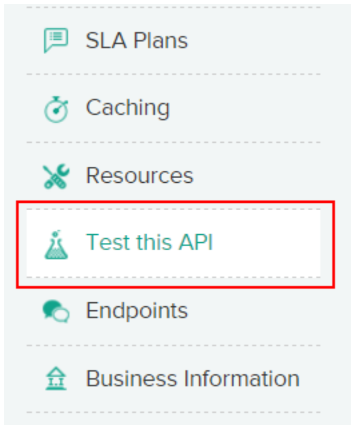
1. Enter the name of the resource in the Endpoint Path field. For example, if your endpoint URL represents an online store, you can define the following resources:
   - /products. For example, http://endpointurl/products
   - /orders. For example, http://endpointurl/orders
   - Nested orders, {orderId}/status. For example, http://endpointurl/{orderId}/status
2. Enter a description of the API in the Nickname field. A nickname is a one word description of the API. The resource uses the nickname as an identifier. For example, the nickname /checkPhone can be assigned to the resource /CheckPhoneNumber.
3. Select an HTTP method in the URL to make a request.
4. To add the new resource, click **Add Resource**. The resource with an HTTP method displays along with the mapped name (the name given to the resource).
5. Click the HTTP method to change the return type, authentication type, rate limits, and scope of the API.

   You can also change the scope of the API resources.
6. Add the query parameters to the API path and modify the properties.
7. To define the output format or response, click **Manage Models**. You can add or modify a response format and create a return type for a resource.

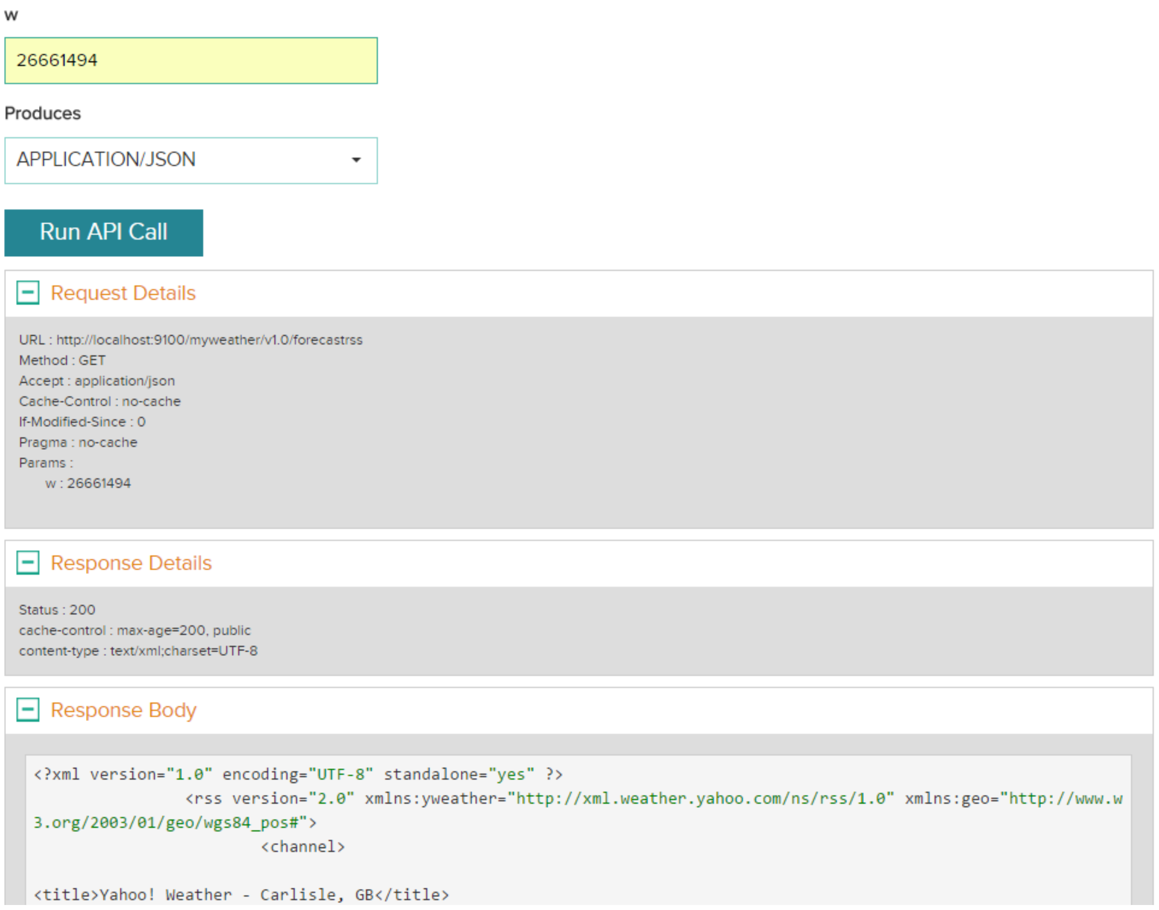


Trying out an API

After creating an API, you can test the API to see the response and other details. Click **Test this API** on the left pane.



Enter the value of the parameter and select the output format. Click **Run API Call**. You can see the response details, as shown below:



```
URL : http://localhost:9100/myweather/v1.0/forecastrss
Method : GET
Accept : application/json
Cache-Control : no-cache
If-Modified-Since : 0
Pragma : no-cache
Params :
    w : 26661494
```

```
Status : 200
cache-control : max-age=200, public
content-type : text/xml;charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
                <rss version="2.0" xmlns:yweather="http://xml.weather.yahoo.com/ns/rss/1.0" xmlns:geo="http://www.w
3.org/2003/01/geo/wgs84_pos#">
                        <channel>

<title>Yahoo! Weather - Carlisle, GB</title>
```

Viewing all APIs

In this page, you can view all APIs you have:

- Published
- Created as draft
- Deprecated
- Retired

Also, you can filter the list of APIs according to API visibilit. For example:

- Public
- Partner
- Intranet

API Catalog

In this page, you can filter all APIs according to the visibility and time (newest or oldest).

Changing the header logo and title

You can change the logo and title on the header area of the administrator portal.

Navigate to the <API Manager Installation Directory>/wwwroot/portal/conf folder and open the fileconfig.json. Modify the following properties:

{

"headerTitle": "API Manager Portal",

"adminHeaderTitle": "API Manager Administrator",

"headerLogoPath": "images/CF-Logo.png"

}

Cross-Origin Resource Sharing (CORS)

CORS allows restricted resources in the API to be accessed from another domain. CORS is a standard mechanism that allows JavaScript calls executed in a web page to interact with resources originating from an external domain. All known browsers enforce this policy.

Allowed HTTP Methods - A list of HTTP methods that are allowed, for example, GET, PUT, POST, and so forth.

Allowed Origins - Represents the origin of a request or *. For a request to be allowed cross-domain, add the header specified in **Allowed Origins** to the request. For example, if the server responds with allowed origin as http://examplerest.com, then only requests from http://examplerest.com are allowed.

Note: The Access-Control-Allow Origin Header should contain the list of origins that can use this API from the browser. The header also allows a wildcard value * adding it would allow any website can invoke this API and retrieve the response. The wild card value can be abused in some cases like where an external website part of internet can be able to consume the API which is part of Intranet zone.

Allowed HTTP Headers - A list of allowable custom request headers, for example, Origin, Content-Type, and so forth.

Allowed Expose Headers - Specify the headers you want to grant explicit permissions so that the client can read those headers. Doing so ensures that the client triggers a cross-origin request.

Preflight Max Age - Specify the amount of time (in seconds) for the browser to cache the response to a request.

Example

For example, you want to GET data from http://api.examplerest.com/user from http://examplerest.com. There is an access token, xyz123, that you can pass in the Authorization header to authenticate the request.

The browser sends an OPTIONS request to the server with the following header:

OPTIONS /user

Origin: http://www.examplerest.com

Access-Control-Request-Method: GET

Access-Control-Request-Headers: Authorization

Since you want to allow the request, you respond to the request with the header:

Allowed-Origin: http://www.examplerest.com

Allowed HTTP Headers: AUTHORIZATION

Allowed HTTP Methods: GET

The browser now sends the GET request to your server where the API is hosted.

Caching

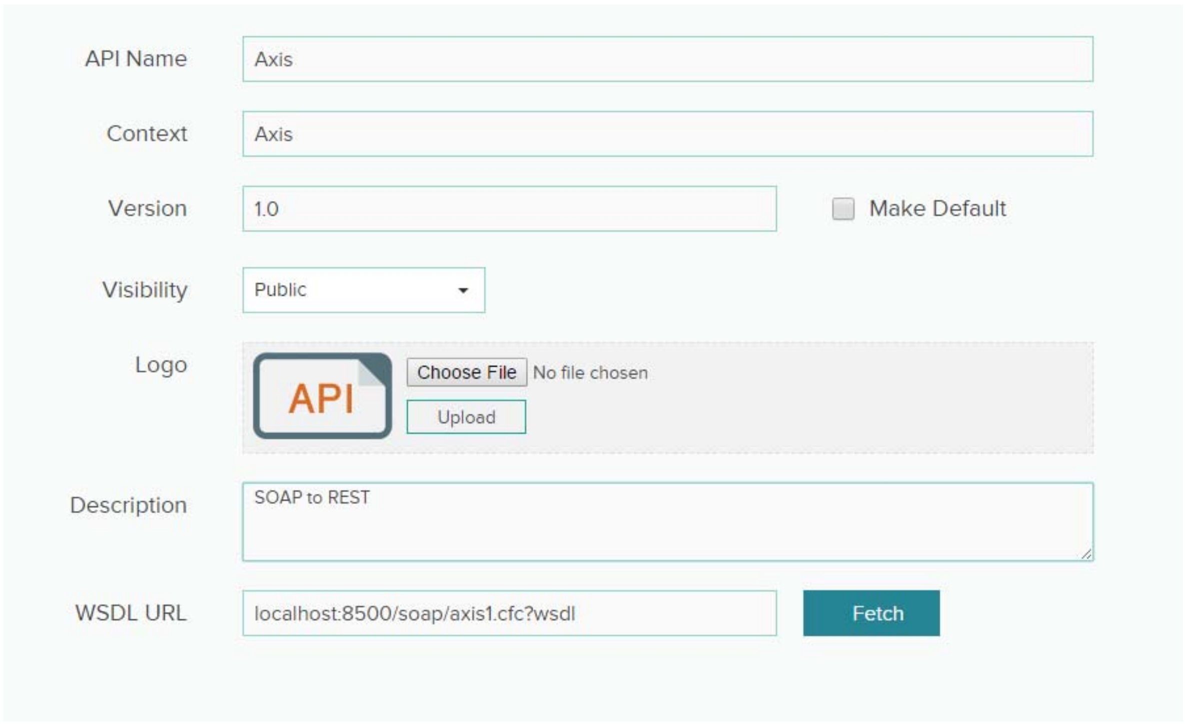
To enable caching of your newly created REST API, click Caching. The benefits of caching an API are speed and reduced load on server. Select the **Enable Caching** check-box and enter the caching timeout in seconds.

Converting SOAP endpoints to REST endpoints

To convert a ColdFusion SOAP service into a REST service, click **Create REST API from SOAP**.

1 In the General Information section, enter the new API information, such as:

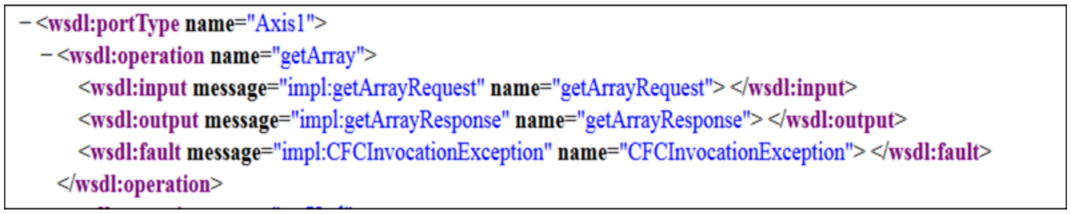
- API Name
- Context
- Visibility
- Version
- Description
- WSDL URL



API Name	Axis
Context	Axis
Version	1.0
Visibility	Public
Description	SOAP to REST
WSDL URL	localhost:8500/soap/axis1.cfc?wsdl

2 In the **WSDL URL** field, enter a SOAP CFC URL.

The example WSDL URL has a single port type. A WSDL <portType> element defines a web service, the operations that can be performed, and the messages that are involved.



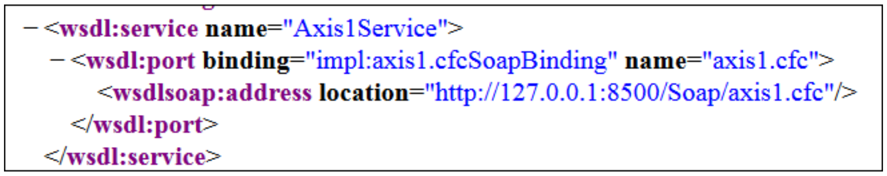
```
- <wsdl:portType name="Axis1">
  - <wsdl:operation name="getArray">
      <wsdl:input message="impl:getArrayRequest" name="getArrayRequest"> </wsdl:input>
      <wsdl:output message="impl:getArrayResponse" name="getArrayResponse"> </wsdl:output>
      <wsdl:fault message="impl:CFCInvocationException" name="CFCInvocationException"> </wsdl:fault>
    </wsdl:operation>
```

This WSDL has a single binding. WSDL bindings define the message format and protocol details for a web service. In this case, axis1.cfcSoapBinding is the binding.

The soap:binding element has two attributes - style and transport. The style attribute can be either rpc or document.

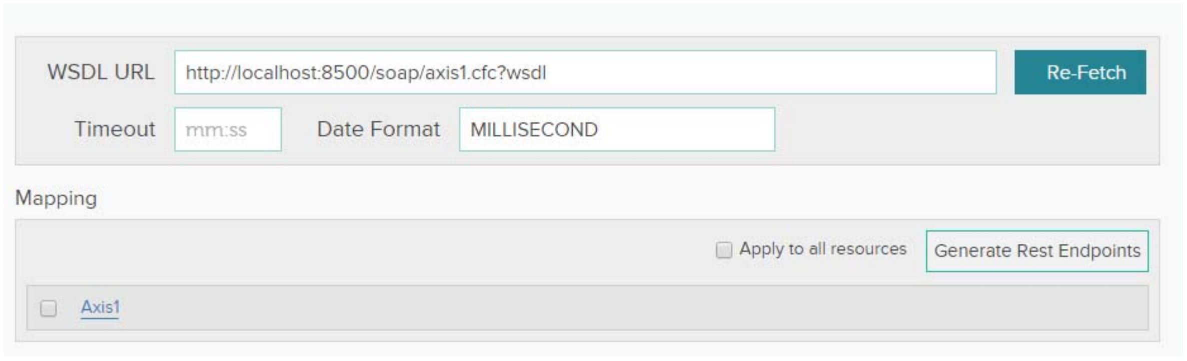
The transport attribute provides a namespace for a SOAP transportation protocol.

For example, **transport="http://schemas.xmlsoap.org/soap/http"** represents SOAP over HTTP.



```
- <wsdl:service name="Axis1Service">
  - <wsdl:port binding="impl:axis1.cfcSoapBinding" name="axis1.cfc">
      <wsdlsoap:address location="http://127.0.0.1:8500/Soap/axis1.cfc"/>
    </wsdl:port>
  </wsdl:service>
```

3 Click **Fetch**. You can see the WSDL resources mapped to REST resources. Choose the port type.



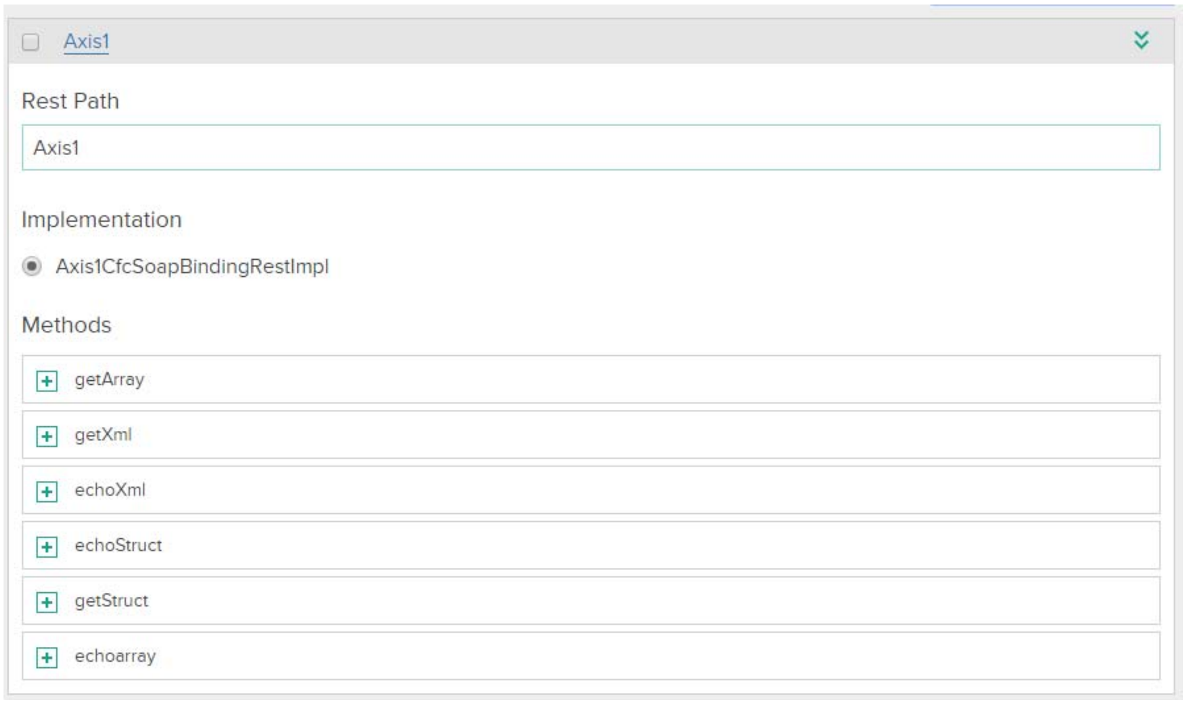
WSDL URL: http://localhost:8500/soap/axis1.cfc?wsdl

Timeout: mm:ss

Date Format: MILLISECOND

Mapping

4 On the WSDL Mapping Wizard, you can view the list of all the operations of the Axis1 port type.



Axis1

Rest Path

Axis1

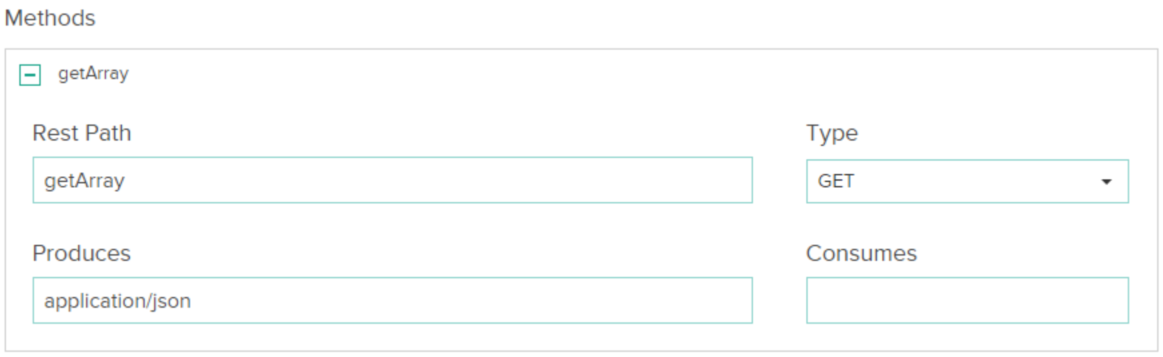
Implementation

● Axis1CfcSoapBindingRestImpl

Methods

- getArray
- getXml
- echoXml
- echoStruct
- getStruct
- echoarray

Expand a method from the list. The REST path of the method displays along with the MIME-type and the method type (GET, PUT, POST, and so forth).



Methods

getArray

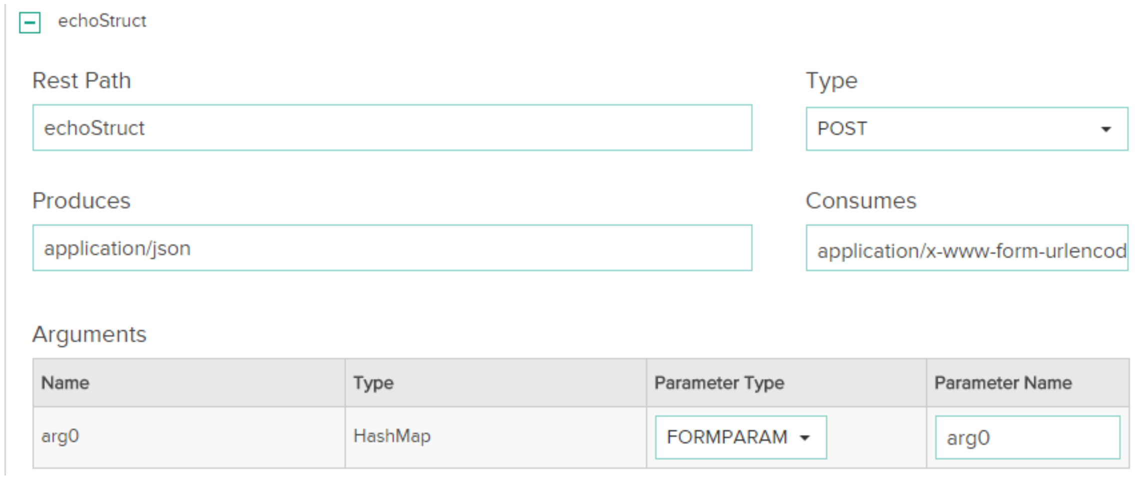
Rest Path: getArray

Type: GET

Produces: application/json

Consumes:

For operations with method type POST or PUT, select the type of parameter for an argument associated with the operation.


echoStruct

Rest Path	Type
echoStruct	POST

Produces	Consumes
application/json	application/x-www-form-urlencod

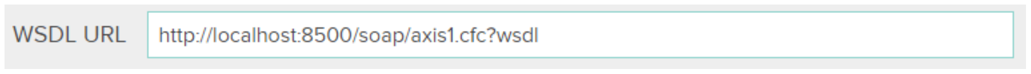
Arguments

Name	Type	Parameter Type	Parameter Name
arg0	HashMap	FORMPARAM	arg0

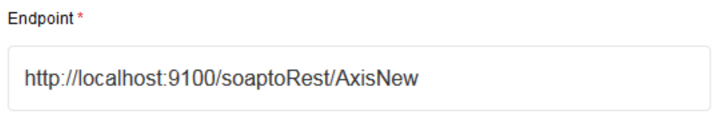
- PATHPARAM describes a path parameter in the URL. For example, in the method GET /directory/{directoryName}, {directoryName} is the PATHPARAM.
- QUERYPARAM describes a parameter of the query string. For example, /search?q=johndoe.
- FORMPARAM inserts web form parameters into a REST service.

5 To generate the REST resources, click **Generate REST Endpoints**. Once you publish, you cannot change the name of the API.

The REST endpoint displays after the conversion. The new URL is:


WSDL URL http://localhost:8500/soap/axis1.cfc?wsdl

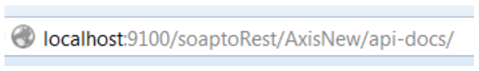
The REST endpoint displays after the conversion. The new URL is:


Endpoint *

http://localhost:9100/soaptoRest/AxisNew

You can use the URL in the following ways:

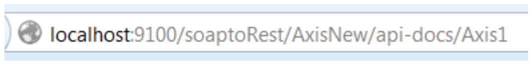
To view the swagger resource listing document, enter the following in the address bar of a browser:


localhost:9100/soaptoRest/AxisNew/api-docs/

You can see the following:

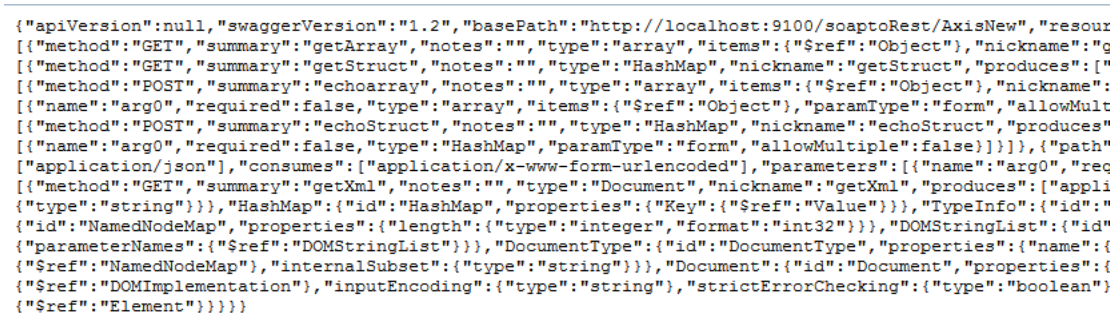
```
{"apiVersion":null,"swaggerVersion":"1.2","apis":[{"path":"/Axis1"}]}
```

To view the root resource path document, enter the following in the address bar:



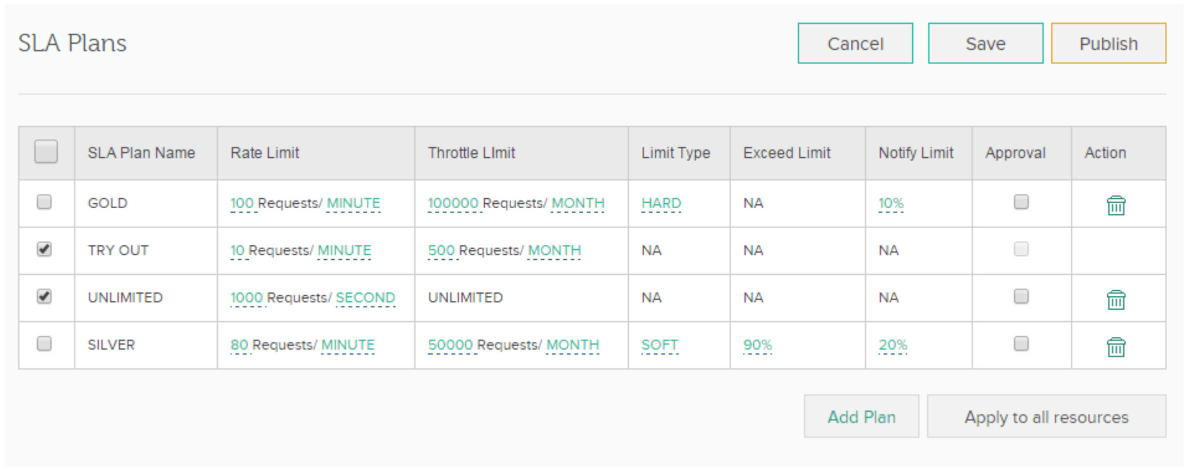
localhost:9100/soaptoRest/AxisNew/api-docs/Axis1

You can see the following:



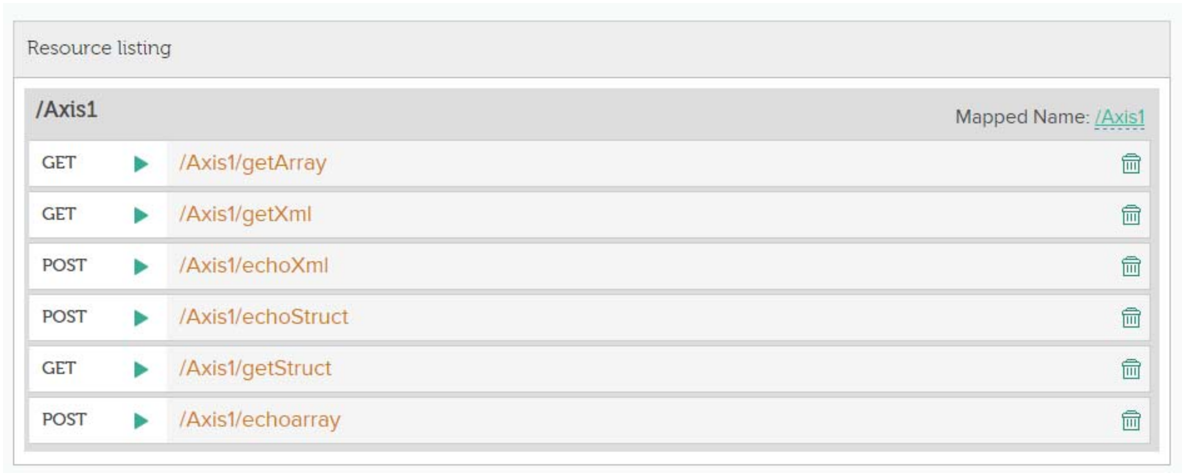
```
{"apiVersion":null,"swaggerVersion":"1.2","basePath":"http://localhost:9100/soaptoRest/AxisNew","resour
[{"method":"GET","summary":"getArray","notes":"","type":"array","items":{"$ref":"Object"},"nickname":"g
[{"method":"GET","summary":"getStruct","notes":"","type":"HashMap","nickname":"getStruct","produces":["
[{"method":"POST","summary":"echoarray","notes":"","type":"array","items":{"$ref":"Object"},"nickname":
[{"name":"arg0","required":false,"type":"array","items":{"$ref":"Object"},"paramType":"form","allowMult
[{"method":"POST","summary":"echoStruct","notes":"","type":"HashMap","nickname":"echoStruct","produces"
[{"name":"arg0","required":false,"type":"HashMap","paramType":"form","allowMultiple":false}]}]},{"path"
["application/json"],"consumes":["application/x-www-form-urlencoded"],"parameters":[{"name":"arg0","req
[{"method":"GET","summary":"getXml","notes":"","type":"Document","nickname":"getXml","produces":["appli
{"type":"string"}}},"HashMap":{"id":"HashMap","properties":{"Key":{"$ref":"Value"}}},"TypeInfo":{"id":"
{"id":"NamedNodeMap","properties":{"length":{"type":"integer","format":"int32"}}},"DOMStringList":{"id"
{"parameterNames":{"$ref":"DOMStringList"}}},"DocumentType":{"id":"DocumentType","properties":{"name":{
{"$ref":"NamedNodeMap"},"internalSubset":{"type":"string"}}},"Document":{"id":"Document","properties":{
{"$ref":"DOMImplementation"},"inputEncoding":{"type":"string"},"strictErrorChecking":{"type":"boolean"}
{"$ref":"Element"}}}}}
```

6 In the SLA Plans section, select a plan or plans for the API.



☐	SLA Plan Name	Rate Limit	Throttle Limit	Limit Type	Exceed Limit	Notify Limit	Approval	Action
☐	GOLD	100 Requests/ MINUTE	100000 Requests/ MONTH	HARD	NA	10%	☐	🗑
☑	TRY OUT	10 Requests/ MINUTE	500 Requests/ MONTH	NA	NA	NA	☐	
☑	UNLIMITED	1000 Requests/ SECOND	UNLIMITED	NA	NA	NA	☐	🗑
☐	SILVER	80 Requests/ MINUTE	50000 Requests/ MONTH	SOFT	90%	20%	☐	🗑

7 In the Resources section, there is also a list of all SOAP operations that are converted to REST. You can see the methods with each operation.


Resource listing

/Axis1 Mapped Name: /Axis1

GET	/Axis1/getArray
GET	/Axis1/getXml
POST	/Axis1/echoXml
POST	/Axis1/echoStruct
GET	/Axis1/getStruct
POST	/Axis1/echoarray

8 Click **Publish** to make the REST API available to subscribers.

Note: *The SOAP to REST feature does not support the WS-* SOAP specifications, for example:*

- WS-Addressing
- WS-Policy
- WS-ResourceProperties
- WS-Security
- WS-Transactions
- WS-ReliableMessaging
- WS-ResourceLifetime

Note: *For some SOAP services, resources that do not get converted to REST out of the box, write your own mechanism to convert to REST.*

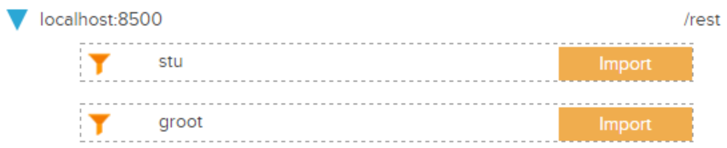
Importing REST API from ColdFusion

Before you import a REST API from ColdFusion server, set up the server. For more information, see ColdFusion and API Manager - Integration.

Set up a CF service that returns the date. In the cfc, use the method GET and a REST path, for example, /GetCurrentDate. Create a folder in the web root of your ColdFusion server and save the cfc.

1 On the Create API screen, click **Import REST API from ColdFusion**.

2 View the list of CF services and click **Import** for the service you want to import from CF server.



3 Once you generate the resources defined in the cfc, set up the SLA plans and the authentication method for the API. To publish the API, click **Publish**.

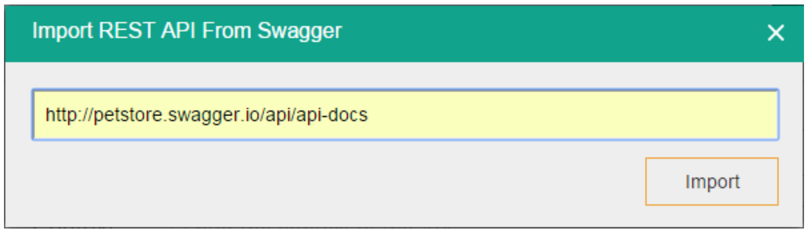
Importing REST API from Swagger

Swagger is a specification and a framework to build API documentation and sandboxes, and to generate the code of an API client.

In ColdFusion API Manager, you can import REST resources based on Swagger specifications and publish them.

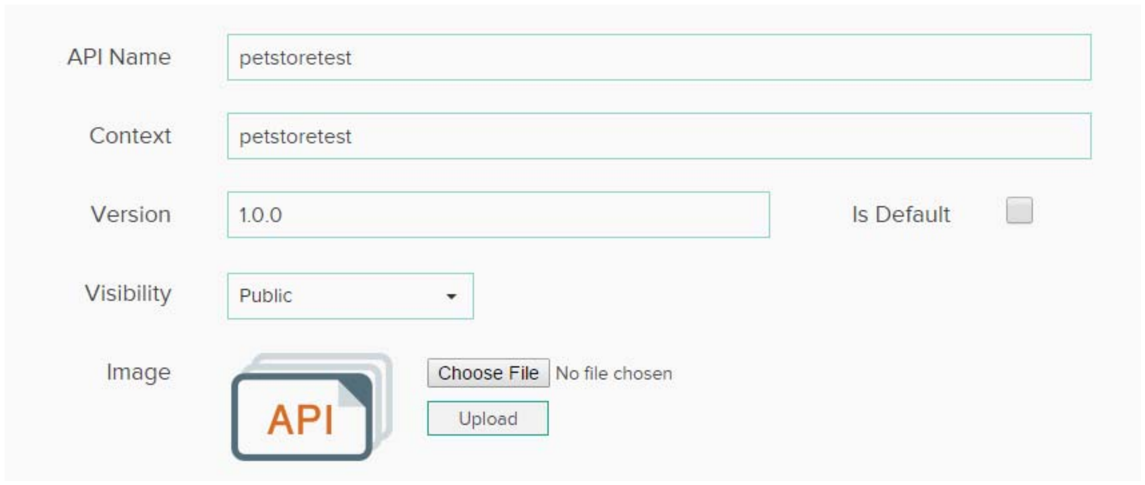
1 On the Create API screen, click **Import REST API from Swagger**.

2 Enter a Swagger API. In this example, use a Petstore server. Click **Import**.

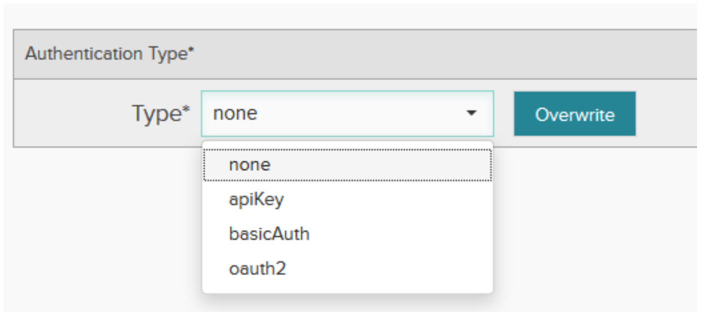


http://petstore.swagger.io/api/api-docs

3 Enter the API information and set the visibility.

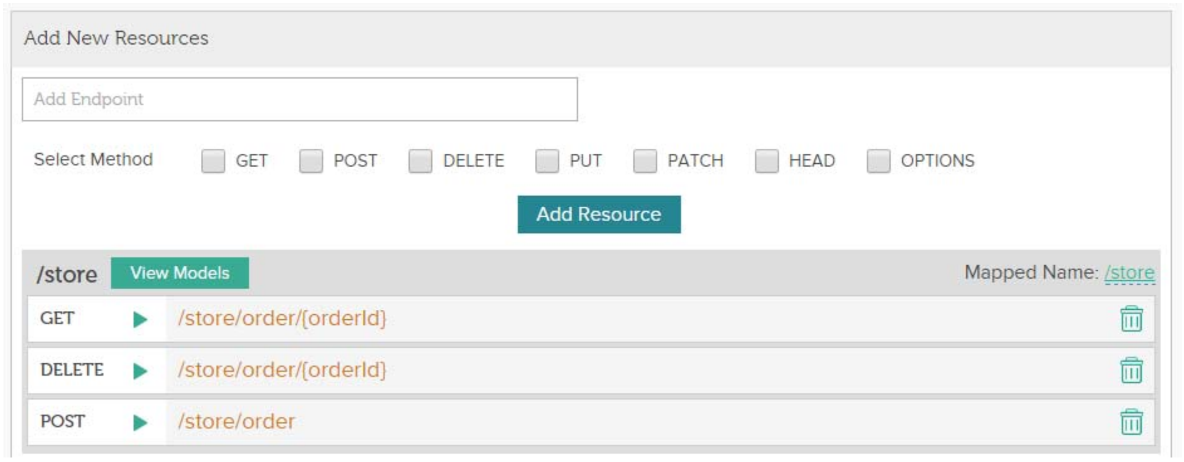

API Name: petstoretest
Context: petstoretest
Version: 1.0.0
Is Default: ☐
Visibility: Public
Image

4 Set the authentication type for the API.



Type*: none

5 To view existing Petstore resources or add a resource, click **Resources** on the left pane on the screen.



6 To publish the API, click **Publish**.

Importing a SOAP API

You can pass a SOAP API through proxy in API Manager. You can:

- Apply security authentication to SOAP web services
- Apply rate limiting
- Apply throttling
- Collect metrics and logging information

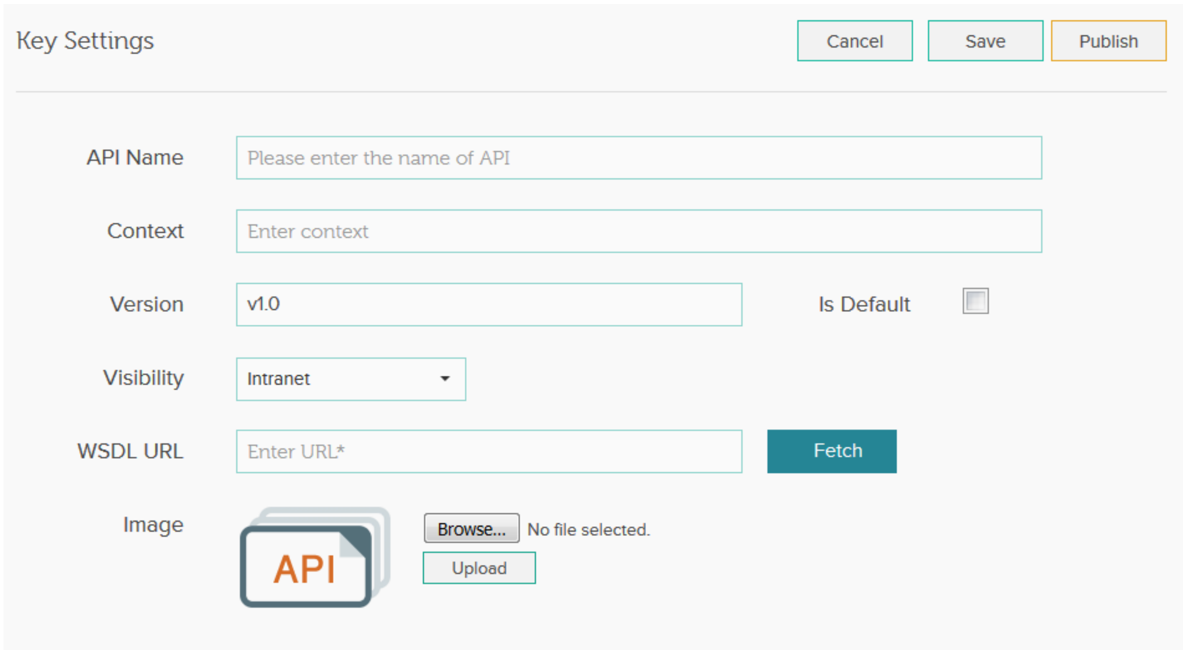
As a publisher, you enter the WSDL URL to a proxy WSDL URL, where the subscriber gets the modified URL with proxy SOAP endpoints and location of the XSD.

An example of proxy WSDL is http://service.adobe.com/<API name>/<version>/?wsdl

If you host a SOAP proxy behind a web server or a particular IP, modify the domain URL through the Administrator portal. The SOAP endpoint is on the domain that the administrator defines in the **Domain URL** field.

The subscriber sends request via proxy endpoint and the request is mapped to the actual endpoint.

1 In the API Manager, click **Import SOAP API**.



Key Settings

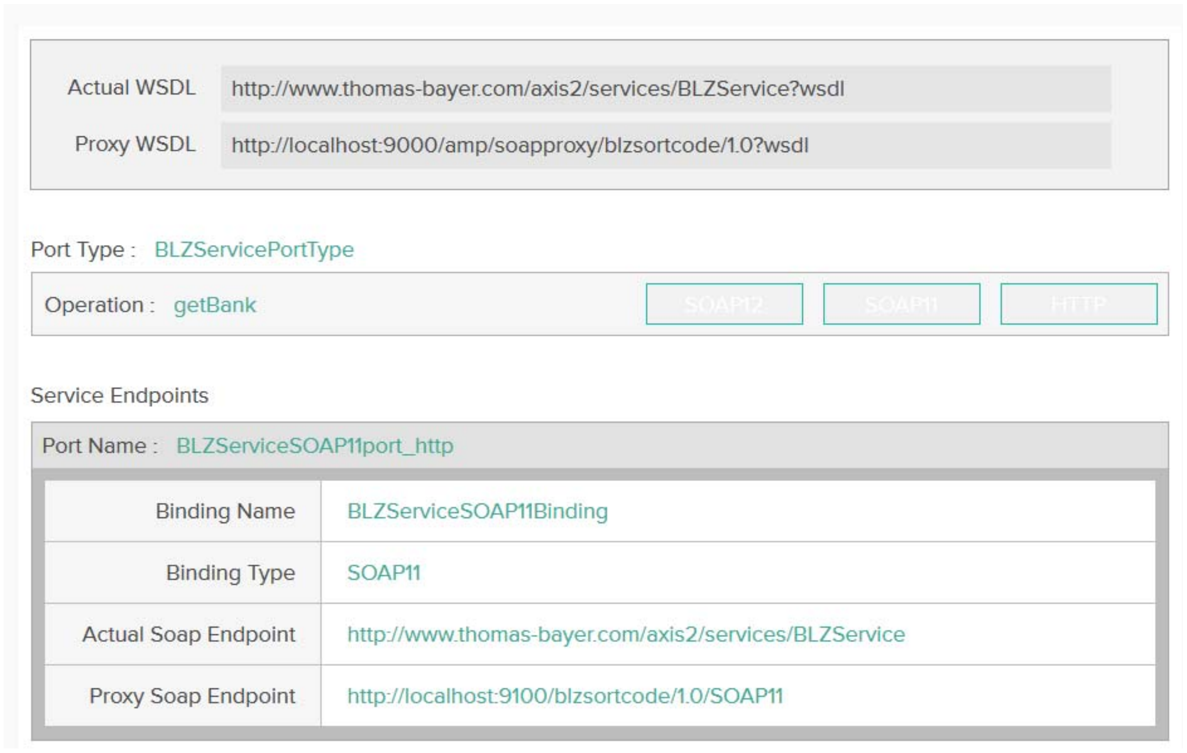
Cancel | Save | Publish

API Name	Please enter the name of API
Context	Enter context
Version	v1.0
Is Default	☐
Visibility	Intranet
WSDL URL	Enter URL*
Image	No file selected.

2 Enter the details on the screen. Set the API visibility to **Public**. Enter a WSDL URL in the **WSDL URL** field. For example, enter the WSDL URL http://www.thomas-bayer.com/axis2/services/BLZService?wsdl.

This WSDL generates request and response for a web service using which you can fetch Bank Sort Codes (called Bank BLZ codes) for banks in Germany.

3 Click **Fetch**. You can see both the actual and proxy WSDLs. You can also see the WSDL ports along with their bindings, and actual and proxy SOAP endpoints. For example,



Actual WSDL	http://www.thomas-bayer.com/axis2/services/BLZService?wsdl
Proxy WSDL	http://localhost:9000/amp/soapproxy/blzsortcode/1.0?wsdl

Port Type : BLZServicePortType

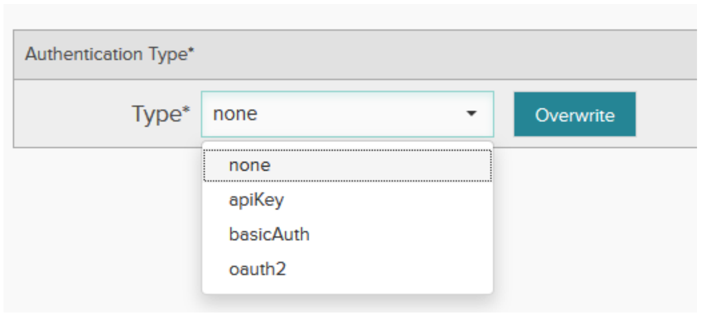
Operation : getBank

Service Endpoints

Port Name : BLZServiceSOAP11port_http

Binding Name	BLZServiceSOAP11Binding
Binding Type	SOAP11
Actual Soap Endpoint	http://www.thomas-bayer.com/axis2/services/BLZService
Proxy Soap Endpoint	http://localhost:9100/blzsortcode/1.0/SOAP11

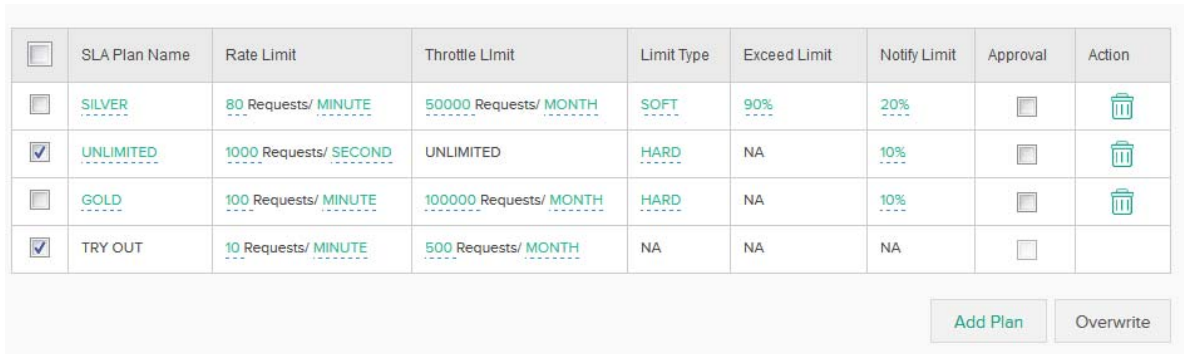
4 Choose the type of authentication for the API from the **Type** drop-down list.



Authentication Type*

Type* none

5 Choose an SLA plan or create a plan. To make the plan subject to approval, select the **Approval** check-box.



☐	SLA Plan Name	Rate Limit	Throttle Limit	Limit Type	Exceed Limit	Notify Limit	Approval	Action
☐	SILVER	80 Requests/ MINUTE	50000 Requests/ MONTH	SOFT	90%	20%	☐	🗑
✓	UNLIMITED	1000 Requests/ SECOND	UNLIMITED	HARD	NA	10%	☐	🗑
☐	GOLD	100 Requests/ MINUTE	100000 Requests/ MONTH	HARD	NA	10%	☐	🗑
✓	TRY OUT	10 Requests/ MINUTE	500 Requests/ MONTH	NA	NA	NA	☐	

When you make an SLA plan ssubject to approval, an API subscriber cannot use the API unless you grant the subscriber approval to use the API under that SLA plan. For example, if you grant a subscriber GOLD plan subject to approval, the subscriber can only subscribe to GOLD if you approve the subscription request.

For more information, see Approving subscriptions .

6 To publish the API and make it available to subscribers, click **Publish**.

Once you publish the API, a subscriber can try out the API and use the API in different applications.

To see how a subscriber can try out the API, see **Try-out SOAP API** .

Authenticating an API

In the API Manager, you can use API keys to authenticate your APIs and applications. The API Manager generates the API keys and enable you to add API key-based authentication to your APIs.

Validation using API keys is a type of security you can enforce while creating an API. Applications use the API key and the API Manager checks to see if the API key is in an approved state for a resource.

The API Manager uses three API authentication types:

1 apiKey
2 basicAuth
3 OAuth2

Authentication types

apiKey

An API Key is an opaque token, which is a simple form of authentication to consume an API. To acquire an API Key, create an application as a subscriber.

Consuming an API protected using API Key

In a request, you can pass an API Key to the API Manager in a request in one of the following ways:

- **Header:** The api_key header can be used to pass the API Key to the API runtime.
- **Query Parameter:** The api_key query parameter can be used to pass the API Key.
- **Form Parameter:** The api_key form parameter can be used to pass the API Key.

To know more about authentication using API Key, see API Key authentication.

basicauth

Basic authentication is the simplest form of authenticating a user by passing the username and password in the request to consume an API. The API Manager allows protecting an API using the Basic authentication scheme.

A publisher can configure the user store to be used for the authentication. Also, if the API resource is protected with a scope configured with roles, then the API Manager also authorizes the role during the consumption of the API call.

In basic authentication, you can enforce authentication according to user stores.

1 From the **User Store** drop-down list, select a user store. The API administrator creates a user store.
2 Create a scope. A scope defines a role defined in a user store. Enter the name of the scope and select a role.
3 To add the newly created scope, click **Add Scope**.

To know more about basicauth authentication, see Basic authentication.

OAuth2

OAuth2 is a standard for delegating authorization to an application on behalf of the user without providing the password. Instead of the user password, an access token having limited lifetime with the limited access (using scopes) will be provided to the application. Depending on the type of application, obtaining an access token varies and are called grant types.

In the API Manager,

- The application requests authorization to access resources in an API you create.
- If you approve the request, the application receives an authorization.
- The application requests an access token after it presents authentication of its identity.
- The API issues an access token to the application.

To set up OAuth2,

1 Select an Oauth type, **USERSTORE** or **SAML**. If you select **USERSTORE**, select a user store defined in **LDAP** or database connection.
2 Add a scope.

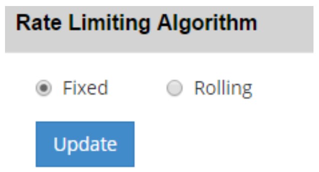
The subscriber creates an application. After you approve the request, the subscriber uses the client secret key to make the API requests.

To know more about OAuth2 authentication, see OAuth2.

Adding tiers to an API

You can enforce limits on each resource consumption and assign multiple consumption levels. You can change the API-throttling level of a plan and add new plans. Throttling in API Manager imposes limits on the maximum rate at which the API Manager can make requests.

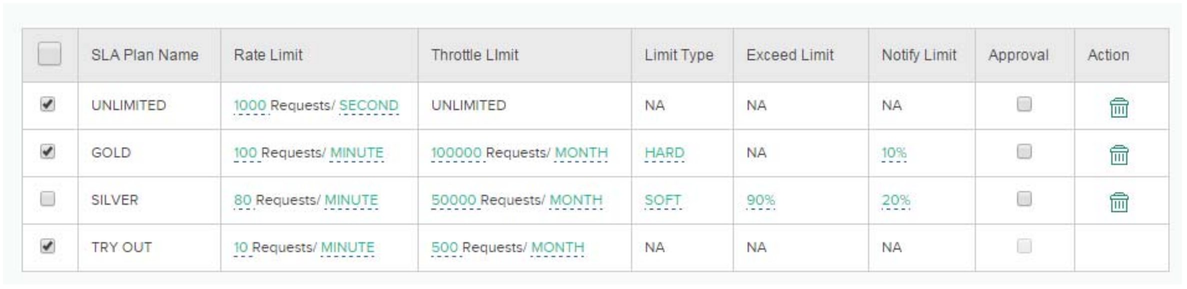
The administrator configures an API Service Level Agreement (SLA) by selecting one of the two algorithms. The administrator then adds an SLA by specifying its rate and throttle limits.


Rate Limiting Algorithm

● Fixed ○ Rolling

As a publisher, you can create a subscription plan and assign throttling limits to an API. To modify an existing tier:

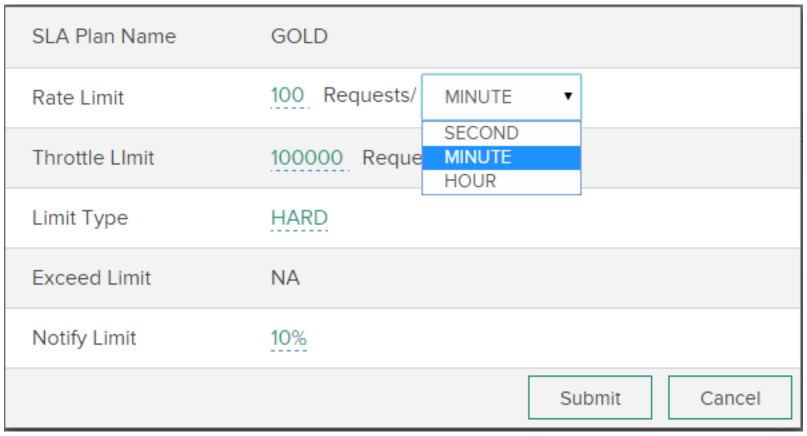
1 Select any plan created by the administrator.



☐	SLA Plan Name	Rate Limit	Throttle LImit	Limit Type	Exceed Limit	Notify Limit	Approval	Action
✓	UNLIMITED	1000 Requests/ SECOND	UNLIMITED	NA	NA	NA	☐	🗑
✓	GOLD	100 Requests/ MINUTE	100000 Requests/ MONTH	HARD	NA	10%	☐	🗑
☐	SILVER	80 Requests/ MINUTE	50000 Requests/ MONTH	SOFT	90%	20%	☐	🗑
✓	TRY OUT	10 Requests/ MINUTE	500 Requests/ MONTH	NA	NA	NA	☐	

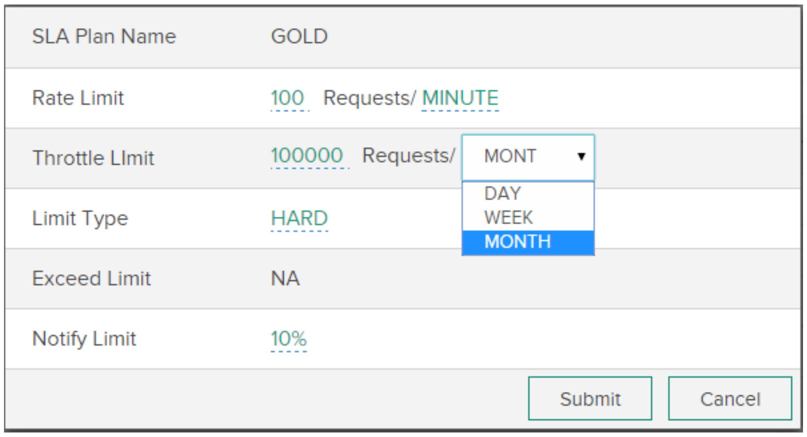
2 Click the name of the plan and change its properties.

3 Modify the API rate limit. You can only limit an API request rate to number of requests per second, minute, or hour.



SLA Plan Name	GOLD
Rate Limit	100 Requests/ MINUTE
Throttle LImit	100000 Reque
Limit Type	HARD
Exceed Limit	NA
Notify Limit	10%

4 Modify the API throttle limit. You can only limit an API throttle rate to number of requests per day, week, or month.

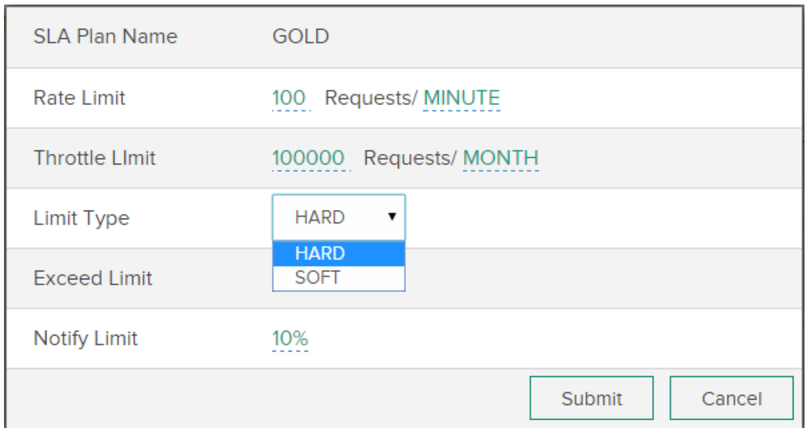


SLA Plan Name	GOLD
Rate Limit	100 Requests/ MINUTE
Throttle LImit	100000 Requests/ MONT
Limit Type	HARD
Exceed Limit	NA
Notify Limit	10%

5 Select the type of limit for the API requests.

There are two options:

- **Hard:** If you select this option, the number of API requests cannot exceed the throttle limit.
- **Soft:** If you select this option, you can set the API request limit exceed a percentage. For example, if you set the Exceed Limit to 90%, the number of API requests can exceed 90% of the prescribed limit.



SLA Plan Name	GOLD
Rate Limit	100 Requests/ MINUTE
Throttle LImit	100000 Requests/ MONTH
Limit Type	HARD
Exceed Limit	
Notify Limit	10%

6 To approve an API request plan, select the **Approval** check-box.

7 To add the new SLA plan, click **Add Plan**.

Adding rate limits to API resources

You can add rate limits to the resources of the API for the SLA plans which the API subscribe. Once the API exceeds the rate limit, the subscriber sees Status 429 in the response header. There is also a try-out rate limit for the subscriber. You do not have any limit to try out your APIs.

To add rate limits to the API resources:

1 Enter the number of requests for an SLA plan.

2 Save the information. The API resources get new rate limits.

Specifying endpoints

An endpoint defines the address to a web service. You can expose both REST and SOAP services to consumers through APIs. Enter the following details in the **Endpoints** section.

Choose the type of enpdpoint.

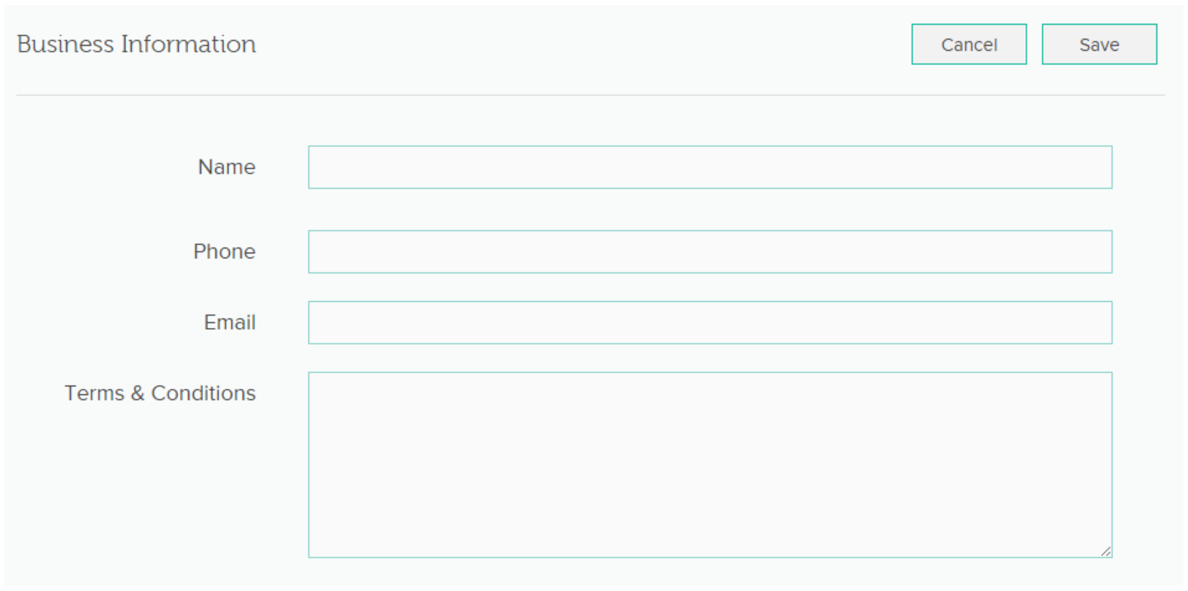
HTTP URL: If you choose this option, enter a primary URL or URLs that contains the resources. You can also enter a test endpoint URL to test your APIs in a sandboxed environment.

Load Balancer: If you choose **Load Balancer**, choose the load balancing algorithm. There are two load balancing algorithms:

- **Round Robin:** In this method, the load is balanced sequentially across the API endpoints. All the endpoints are assigned similar load and have similar performance. Click **Add Another Endpoint** to add the endpoint URLs for load distribution.
- **Weighted Round Robin:** In this method, the load is balanced across the endpoints according to a "weightage" factor that you can assign to each endpoint. The load is divided in terms of the weights assigned. If the weights assigned to the three endpoints are 100, 50, and 50 respectively, the first endpoint handles twice as many requests as the second and third endpoints.

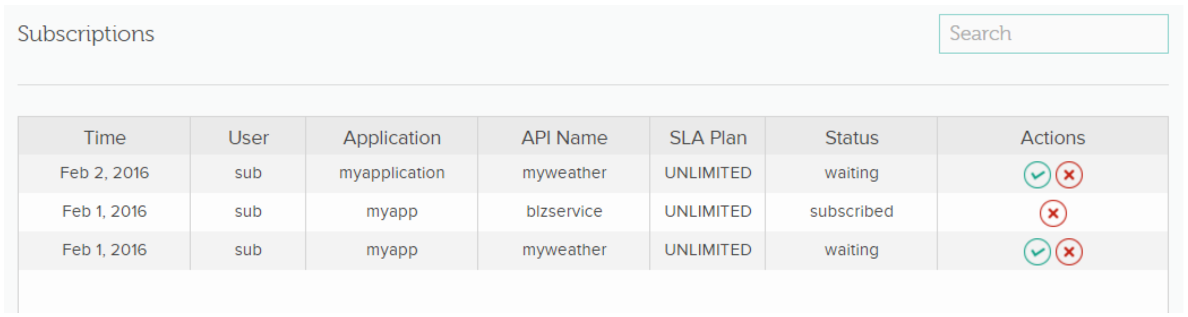
Adding the API business details

Enter the details of your business in the Business Info section.



Approving subscriptions to an API

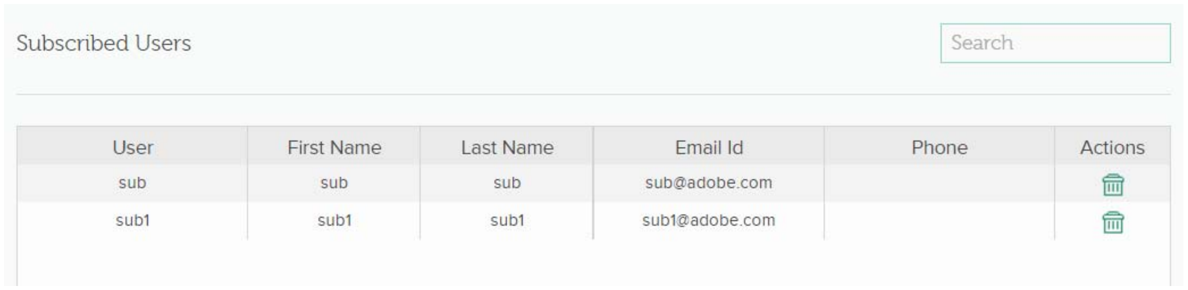
You can approve or reject subsccriptions to an API. Click Subscriptions and view the list all subscription requests.



Time	User	Application	API Name	SLA Plan	Status	Actions
Feb 2, 2016	sub	myapplication	myweather	UNLIMITED	waiting	✓ ✗
Feb 1, 2016	sub	myapp	blzservice	UNLIMITED	subscribed	✗
Feb 1, 2016	sub	myapp	myweather	UNLIMITED	waiting	✓ ✗

Deleting a subscriber

You can delete a subscribed user in the Subscribed Users page. Click the trash button under Actions to delete a subscriber.


Subscribed Users

Search

User	First Name	Last Name	Email Id	Phone	Actions
sub	sub	sub	sub@adobe.com		🗑
sub1	sub1	sub1	sub1@adobe.com		🗑

Viewing subscribed users

View a list of all subscribed users. You can see the following for each user:

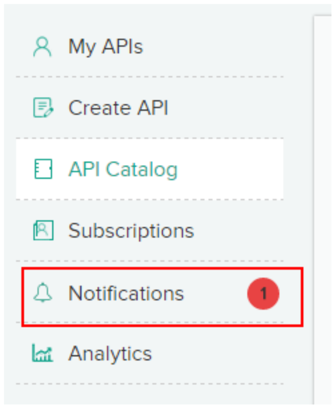
- User name
- First and last name
- Email id
- Phone
- Action

Viewing notifications

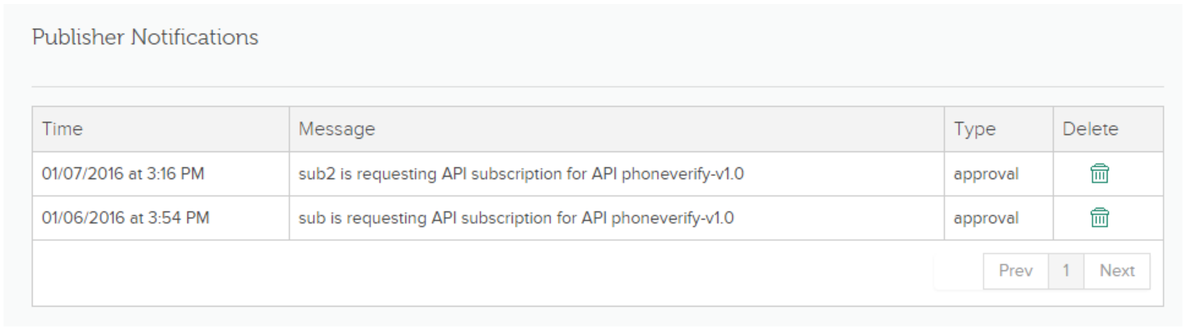
On the Publisher portal, you can check for notifications for scenarios, such as:

- Request to approve API subscription
- Approved API subscription
- Termination or deprecation of an API
- Errors in API endpoints
- An SLA reaches its limits

When a subscriber subscribes to an application, you get a corresponding notification.



When you click **Notifications**, you can view all subscriber messages and take appropriate action.



Publisher Notifications

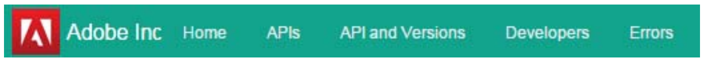
Time	Message	Type	Delete
01/07/2016 at 3:16 PM	sub2 is requesting API subscription for API phoneverify-v1.0	approval	🗑
01/06/2016 at 3:54 PM	sub is requesting API subscription for API phoneverify-v1.0	approval	🗑

Publishing the API

Click **Publish** to publish the API.

Viewing the analytics dashboard

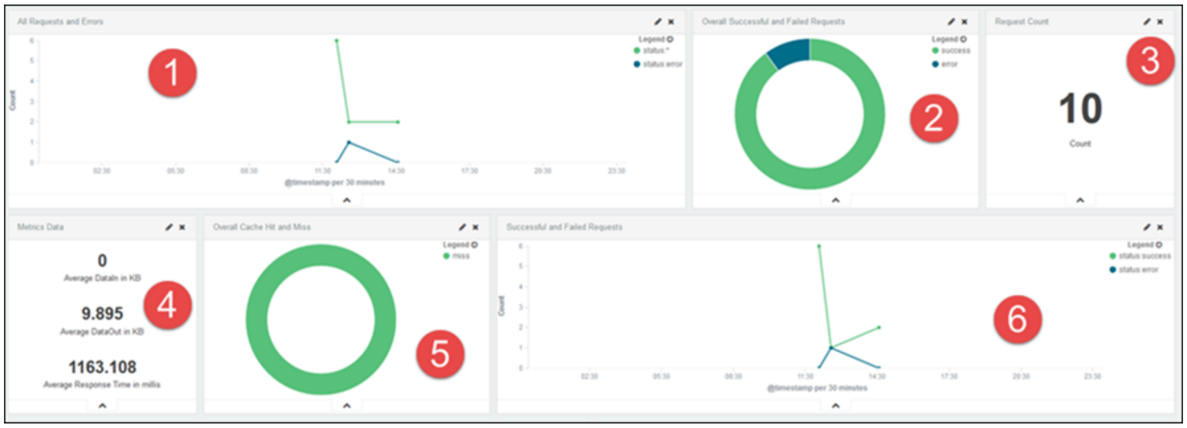
The analytics dashboard for a Publisher consists visualizations for API metrics, successful and failed requests, number of API requests, and so on. Click Analytics to launch the Publisher analytics page.



There are five types of dashboards in the Publisher portal:

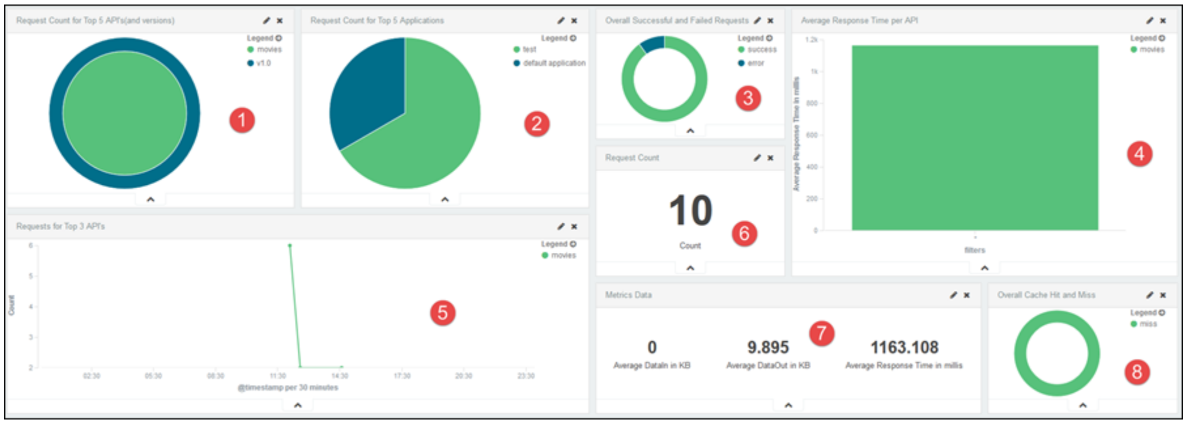
- Home
- APIs
- APIs and Versions
- Developers
- Errors

Home: On the Home dashboard panel, you can see the following visualizations:


All Requests and Errors
1
@timestamp per 30 minutes
Overall Successful and Failed Requests
2
Legend
success
error
Request Count
3
10
Count
Metrics Data
0
Average DataIn in KB
9.895
Average DataOut in KB
1163.108
Average Response Time in millis
4
Overall Cache Hit and Miss
miss
5
Successful and Failed Requests
status success
status error
6
@timestamp per 30 minutes

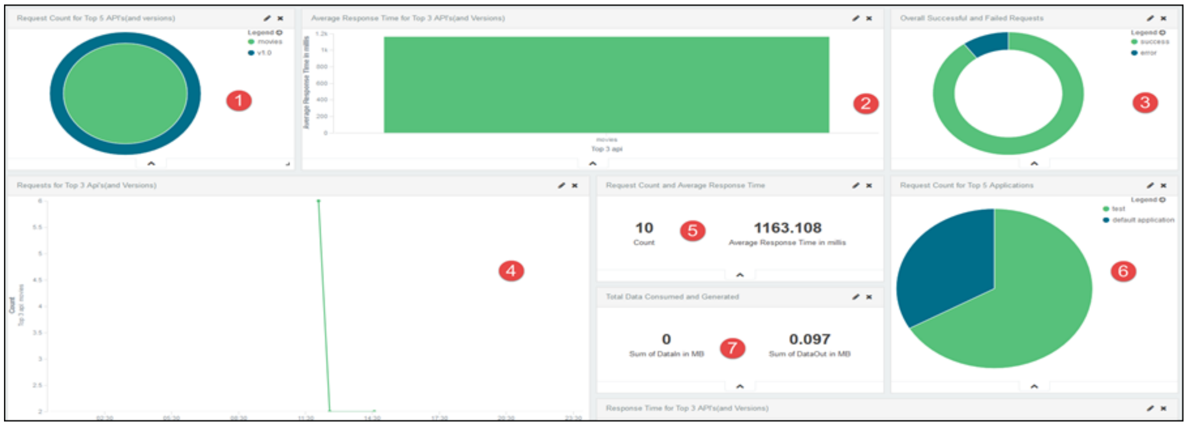
1	The number of API requests within a time range with total requests and erroneous requests.
2	A donut-chart representation of API success and errors.
3	The metrics data like average response time, number of API requests for APIs, and so forth.
4	The average response time of an API in milliseconds.
5	Cache hit or miss.
6	Line chart of successful and unsuccessful API requests.

APIs: On the APIs dashboard panel, you can see the following visualizations:



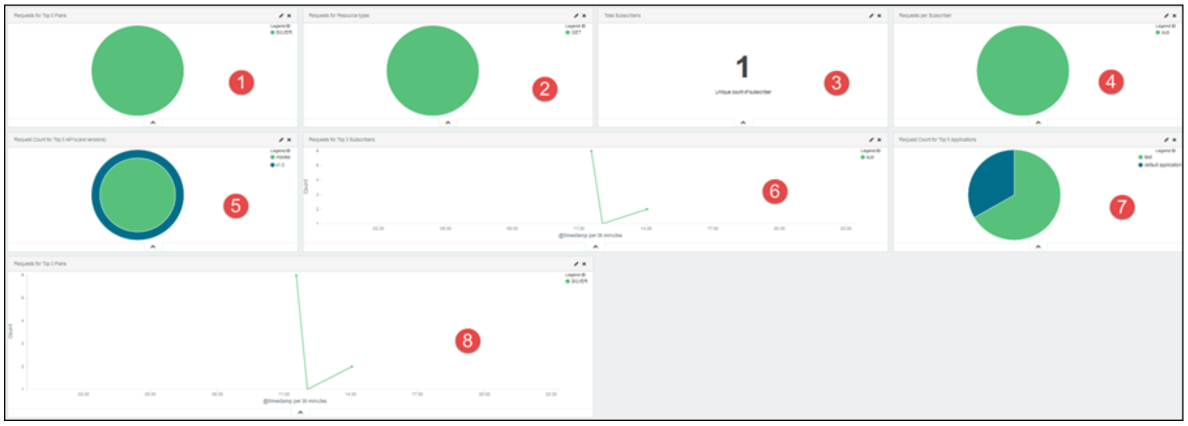
1	The number of requests for the top five APIs and the versions.
2	Donut-chart for the number of API requests by the top five applications.
3	Donut-chart for the percentage of successful and unsuccessful API requests with comparison of the versions of an API.
4	Bar-chart for the average response time for an API.
5	Line-chart for the number of requests by the top three APIs.
6	The number of API requests.
7	Donut-chart of the API metrics data, for example, average response time
8	Representation for cache hit or miss.

APIs and Versions: On the APIs and Versions dashboard panel, you can see the following visualizations:


Request Count for Top 5 API's(and versions)
Average Response Time for Top 3 API's(and Versions)
Overall Successful and Failed Requests
Requests for Top 3 Api's(and Versions)
Request Count and Average Response Time
10
Count
1163.108
Average Response Time in millis
Request Count for Top 5 Applications
Total Data Consumed and Generated
0
Sum of DataIn in MB
0.097
Sum of DataOut in MB
Response Time for Top 3 API's(and Versions)
1
2
3
4
5
6
7

1	The number of requests for the top five APIs and their versions.
2	Bar-chart for the average response times for the top five APIs.
3	Donut-chart for the percentage of successful and unsuccessful API requests.
4	Line-chart for the top three API requests with comparisons of different API versions.
5	The number of API requests and the average response time for the APIs.
6	The number of API requests for the top five applications.
7	API data consumption in MB.

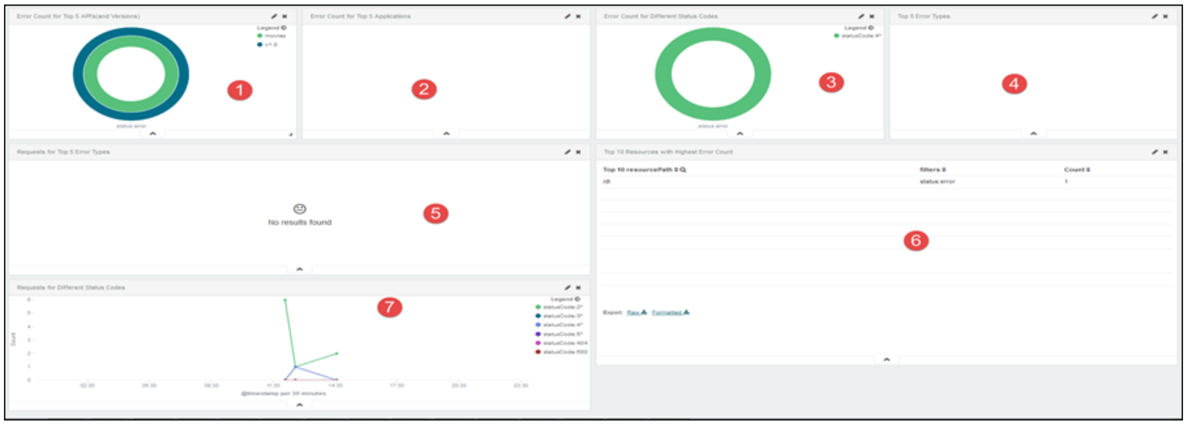
Developers: On the Developers dashboard panel, you can see the following visualizations:



1	Pie-chart for the top five plan usage.
2	The number of API requests for the method types.
3	The number of subscribers of an API.

4	Pie-chart for the number of requests for a subscriber.
5	The number of requests for the top five APIs.
6	Line-chart for the requests from the top three subscribers.
7	Pie-chart for the number of requests for top five applications.
8	Line-chart for requests from top three SLA plans.

Errors: On the Errors dashboard panel, you can see the following visualizations:



1	The number of errors from the top five APIs.
2	The number of errors from the top five applications.
3	Donut-chart for the number of errors from all status codes.
4	Pie-chart for the top five error types.
5	Line-chart for requests for the top five error types.
6	Line-chart for requests for all status codes.
7	List of top ten resources with the maximum number of errors.

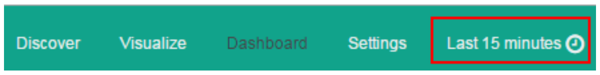
Filtering data according to time range

In the Analytics page, you can filter the results according to a time range. There are three ways options:

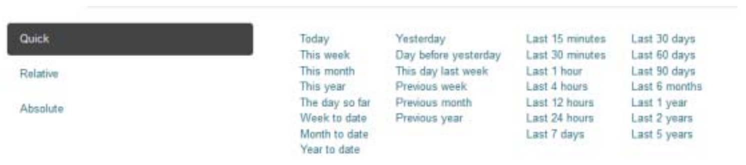
1. Quick
2. Relative
3. Absolute

To filter the data according to time range:

1. Click the time filter as shown below:

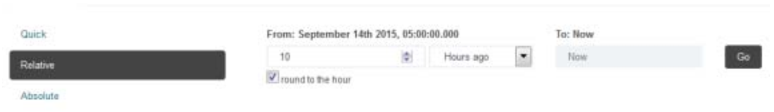


2 Select any time range from the list, as shown below:

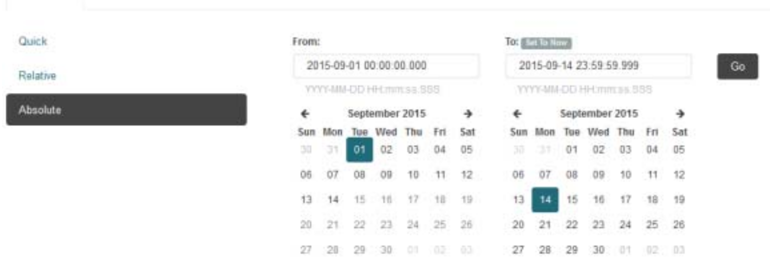


If you select Today, you can see the analytics of all APIs for the last 24 hours.

3 Click Relative. You can filter the data from a specified date and time to the current date and time.



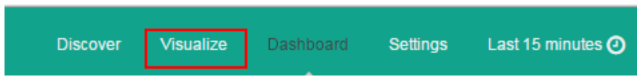
4 Click Absolute. You can filter the results according to dates.



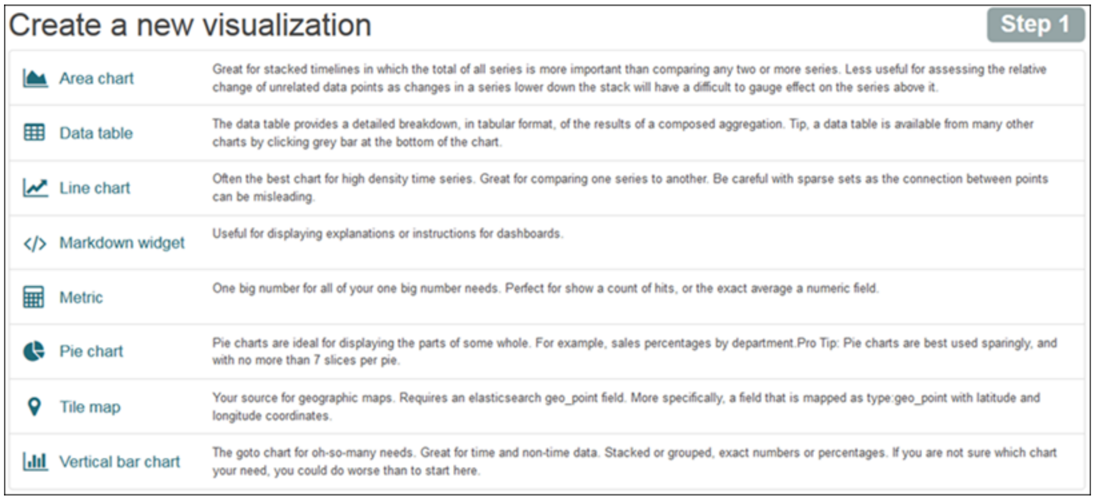
Creating a visualization

You can create your custom visualizations and save it for further analysis. Follow the steps below to create a visualization:

1 To create a visualization, click the Visualize tab.



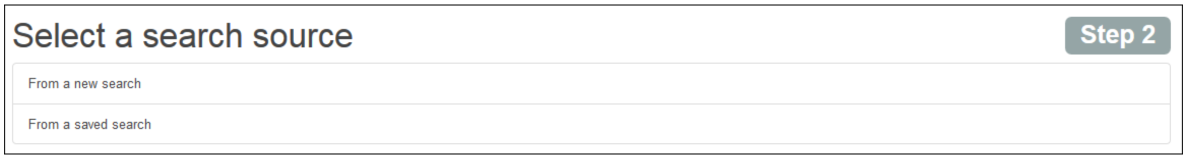
2 You can create visualizations in the form of area charts, data table, pie charts, and so on, from API information. As an example, create a pie chart from the API analytics. Click Pie chart from the list.



Create a new visualization — Step 1

Area chart	Great for stacked timelines in which the total of all series is more important than comparing any two or more series. Less useful for assessing the relative change of unrelated data points as changes in a series lower down the stack will have a difficult to gauge effect on the series above it.
Data table	The data table provides a detailed breakdown, in tabular format, of the results of a composed aggregation. Tip, a data table is available from many other charts by clicking grey bar at the bottom of the chart.
Line chart	Often the best chart for high density time series. Great for comparing one series to another. Be careful with sparse sets as the connection between points can be misleading.
Markdown widget	Useful for displaying explanations or instructions for dashboards.
Metric	One big number for all of your one big number needs. Perfect for show a count of hits, or the exact average a numeric field.
Pie chart	Pie charts are ideal for displaying the parts of some whole. For example, sales percentages by department.Pro Tip: Pie charts are best used sparingly, and with no more than 7 slices per pie.
Tile map	Your source for geographic maps. Requires an elasticsearch geo_point field. More specifically, a field that is mapped as type:geo_point with latitude and longitude coordinates.
Vertical bar chart	The goto chart for oh-so-many needs. Great for time and non-time data. Stacked or grouped, exact numbers or percentages. If you are not sure which chart your need, you could do worse than to start here.

3 Select from the following:

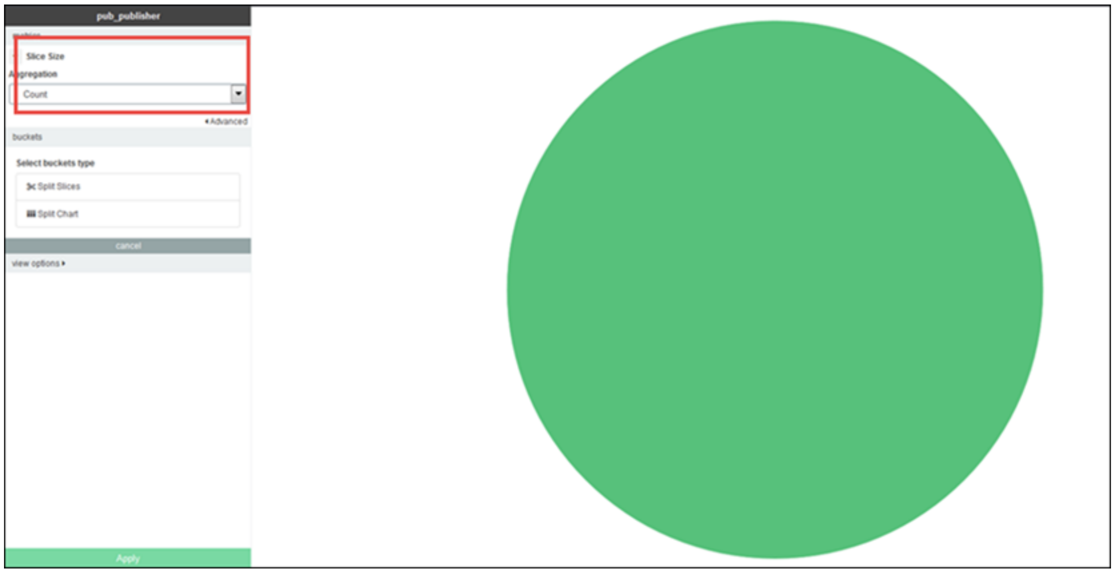


Select a search source — Step 2

From a new search

From a saved search

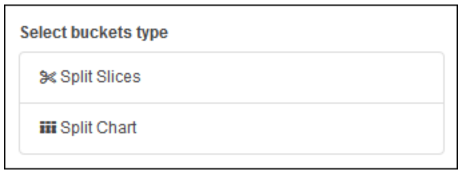
4 Create a visualization from a new search. Select From a new search. You can see a pie-chart representaion of the number of new APIs by a publisher.



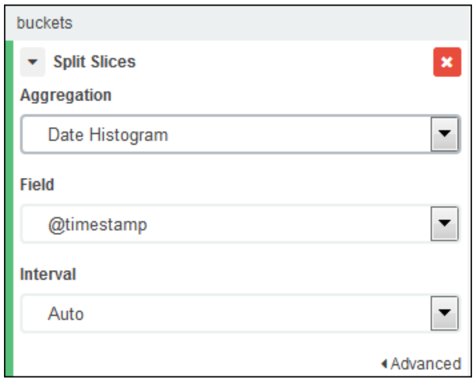
5 Select the type of pie-chart from the following:

- Split Slices

- Split Chart

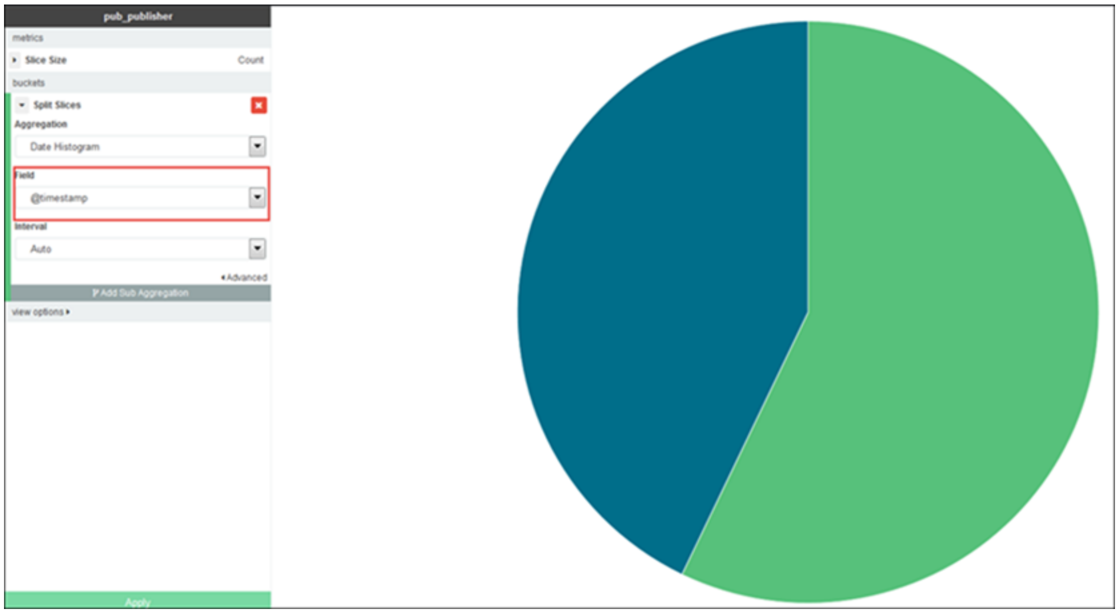


6 Select Split Slices and select Date Histogram from the Aggregation drop-down list.



Aggregation: Date Histogram
Field: @timestamp
Interval: Auto

7 Click Apply. A split pie-chart displays according to the parameters, which in this case is **timestamp**.



Aggregation: Date Histogram
Field: @timestamp
Interval: Auto
Advanced

Subscriber

The API Manager-Subscriber portal enables an API subscriber perform the following:

- View all published APIs
- View all subscribed APIs
- Search for APIs and view them according to category
- View the parameters that an API requires
- Generate an API key
- Choose an SLA tier
- Create applications
- View notifications from publisher

Logging in

To log in to the subscriber portal:

1 Enter <localhost or IP address>:<port number>/portal/ in the address bar of a web browser.

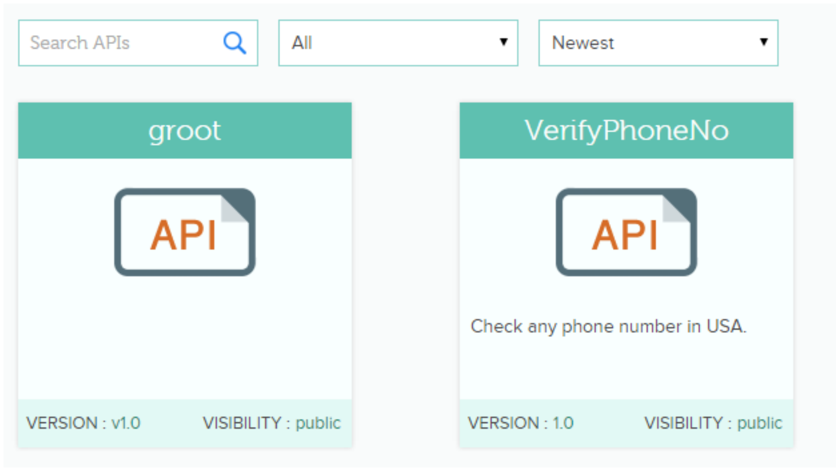
2 Enter the user name and password.

3 Click **Login**.

Browse APIs

After successfully logging in to the portal, you can view the list of subscribed APIs.

1 Choose any API.

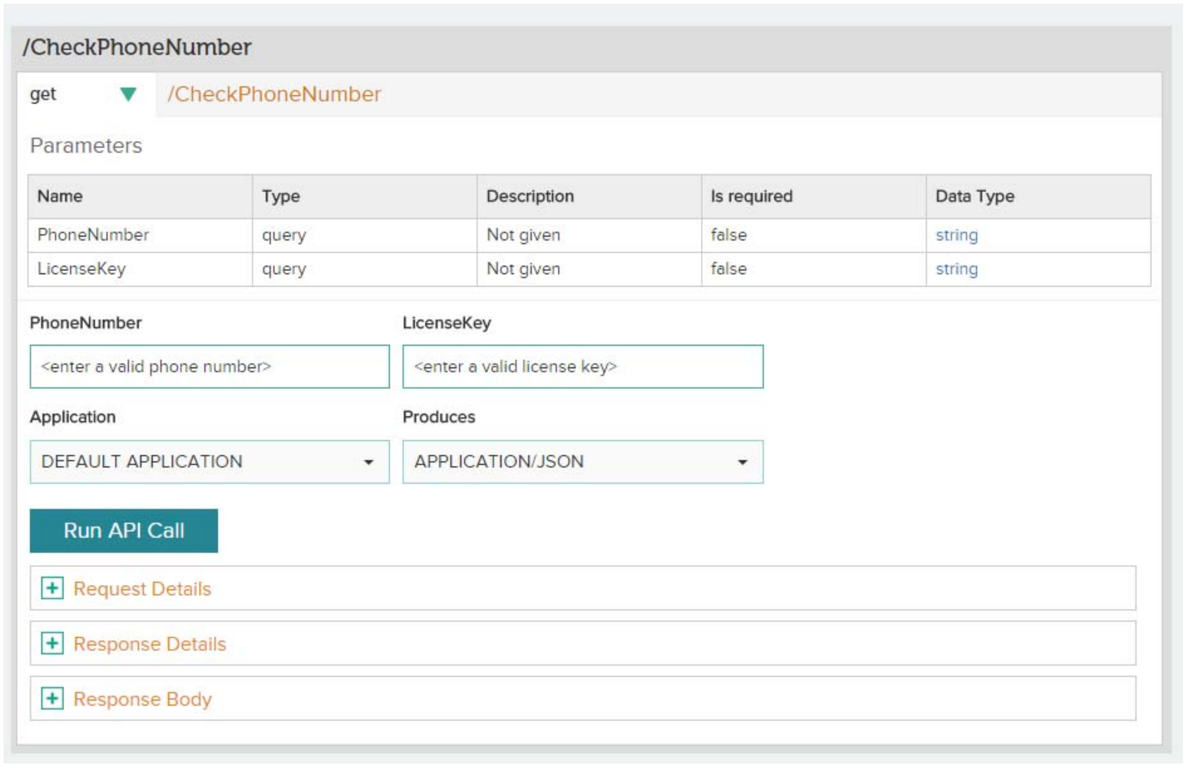


2 On the API Details page, you can view the description of the subscribed API, including:

- Name and description of the API
- Version of the API
- Visibility of the API
- Context of the API

- Lifecycle of the API

3 Choose any API. To try out the API, click Try Out. Select the application and the response content type. Click Run API call. You can see the following results.



/CheckPhoneNumber

get /CheckPhoneNumber

Parameters

Name	Type	Description	Is required	Data Type
PhoneNumber	query	Not given	false	string
LicenseKey	query	Not given	false	string

PhoneNumber: <enter a valid phone number>

LicenseKey: <enter a valid license key>

Application: DEFAULT APPLICATION

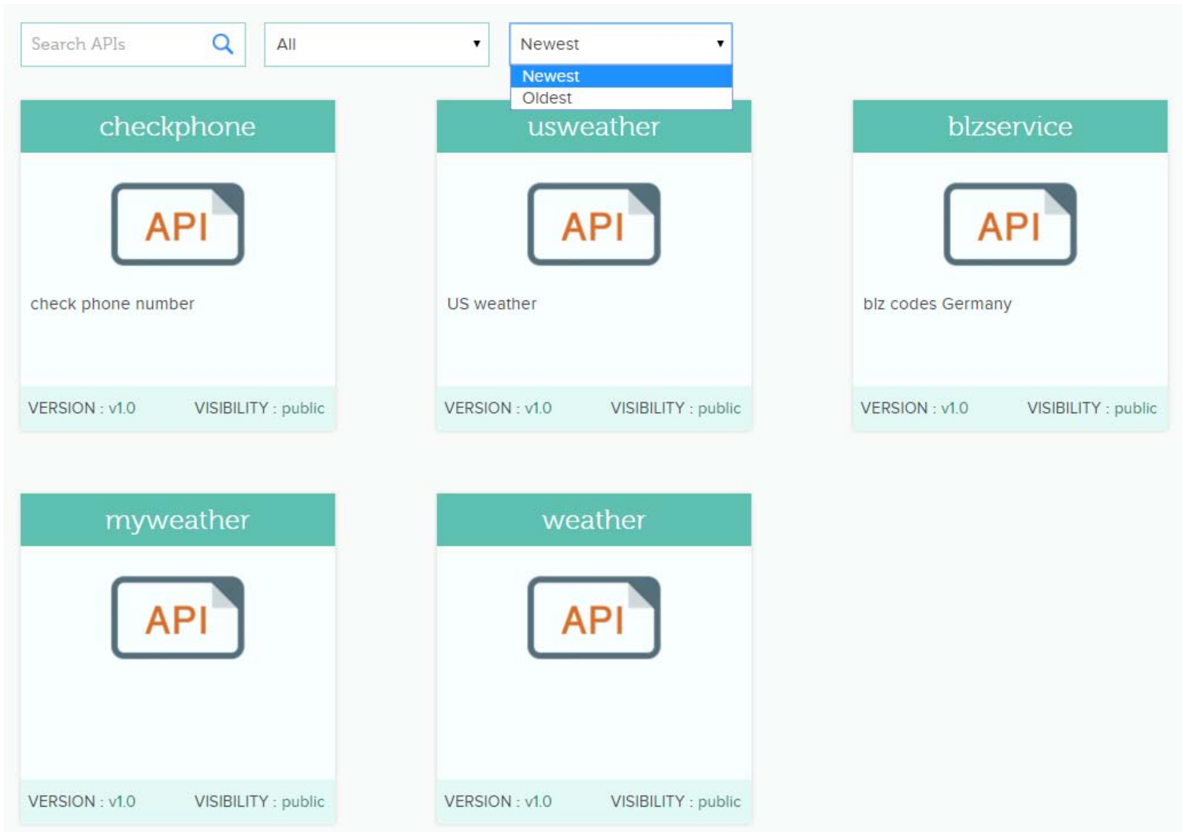
Produces: APPLICATION/JSON

Viewing all APIs

In this page, you can view all APIs you have subscribed to.

API Catalog

In this page, you can filter all subscribed APIs according to the visibility and time (newest or oldest). You can view the APIs along with their version, visibility, description, and so on.



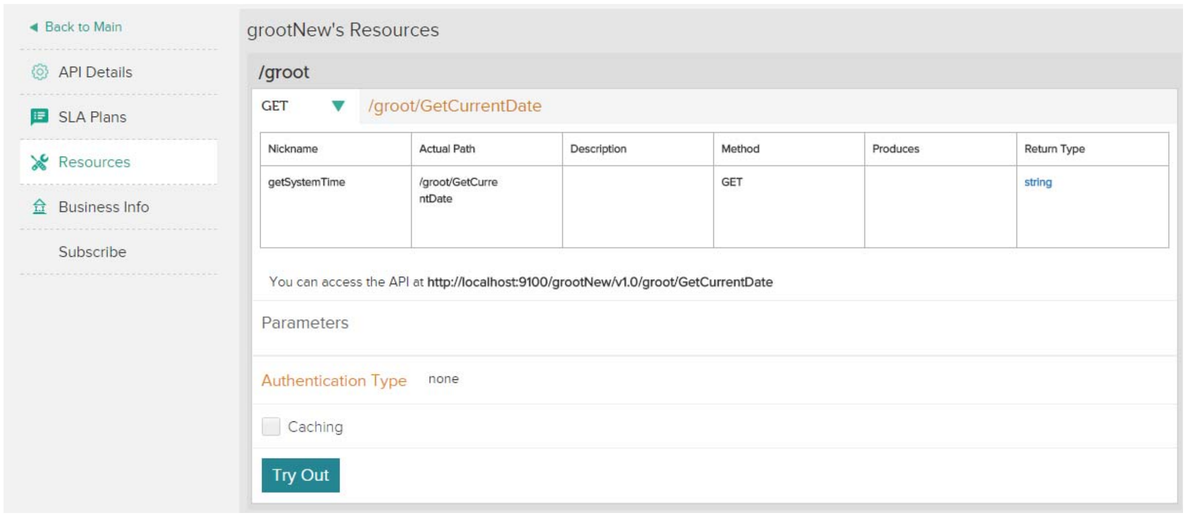
checkphone	usweather	blzservice
check phone number	US weather	blz codes Germany
VERSION : v1.0 VISIBILITY : public	VERSION : v1.0 VISIBILITY : public	VERSION : v1.0 VISIBILITY : public
myweather	weather	
VERSION : v1.0 VISIBILITY : public	VERSION : v1.0 VISIBILITY : public	

Using a REST API from ColdFusion server

Once a publisher creates an API from a ColdFusion server, you can consume and test the API.

1 Choose the API from the API catalog.

2 Click **Resources** on the left panel to view the API details.



grootNew's Resources

/groot

GET /groot/GetCurrentDate

Nickname	Actual Path	Description	Method	Produces	Return Type
getSystemTime	/groot/GetCurrentDate		GET		string

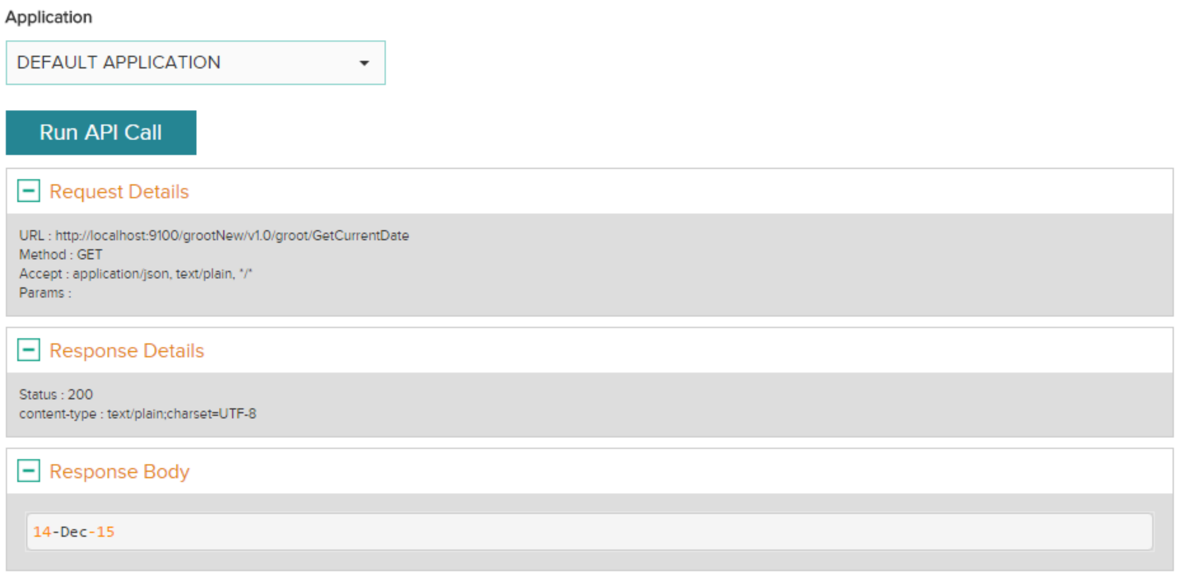
You can access the API at http://localhost:9100/grootNew/v1.0/groot/GetCurrentDate

Parameters

Authentication Type none

Caching

3 To test the API, click **Try Out**. Choose an application and click **Run API Call**. You can see the API returns the current date. Expand the **Request Details** and **Response Details** tabs to view the URL and status code.



```
URL : http://localhost:9100/grootNew/v1.0/groot/GetCurrentDate
Method : GET
Accept : application/json, text/plain, */*
Params :
```

```
Status : 200
content-type : text/plain;charset=UTF-8
```

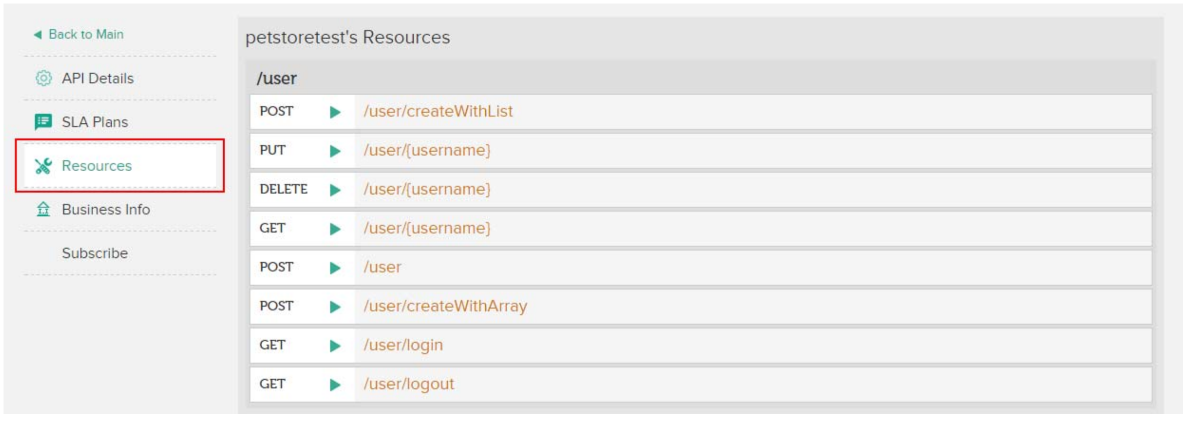
```
14-Dec-15
```

Consuming a Swagger API

Once an API publisher publishes a Swagger API, you can test the API and use the API in your applications. Go to the API catalog to view a list of all public APIs you can use.

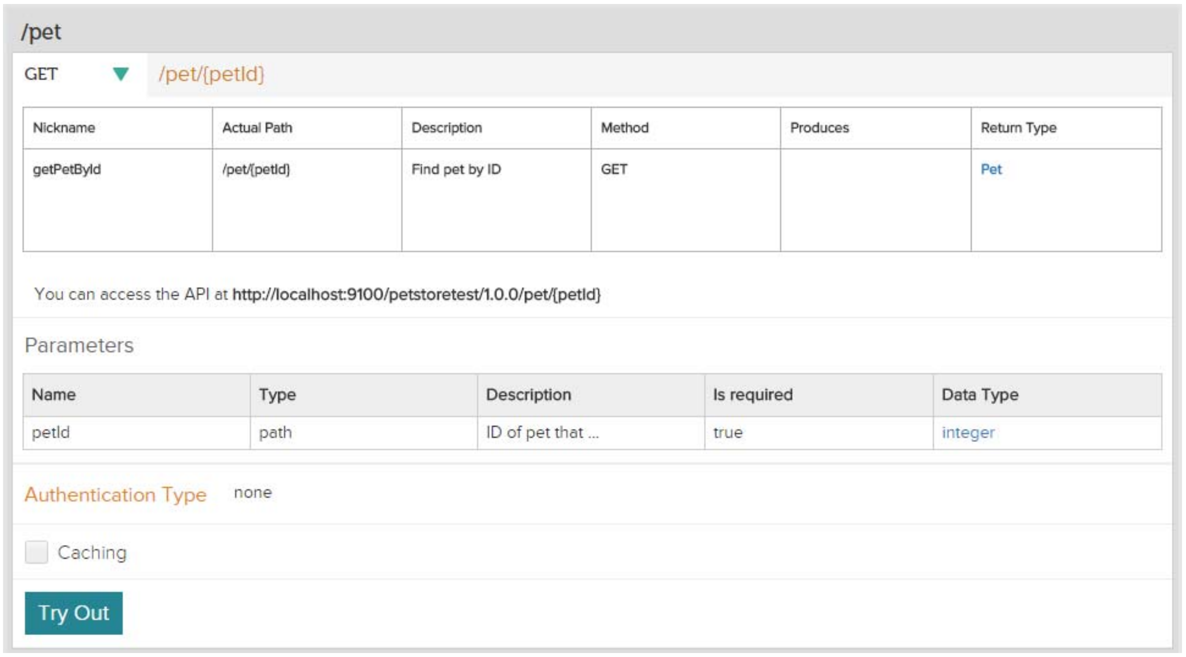
1 Choose the API from the API catalog.

2 Click **Resources** on the left panel to view the API details.



POST	/user/createWithList
PUT	/user/{username}
DELETE	/user/{username}
GET	/user/{username}
POST	/user
POST	/user/createWithArray
GET	/user/login
GET	/user/logout

3 Choose a method for a resource to test the API.


/pet

GET /pet/{petId}

Nickname	Actual Path	Description	Method	Produces	Return Type
getPetById	/pet/{petId}	Find pet by ID	GET		Pet

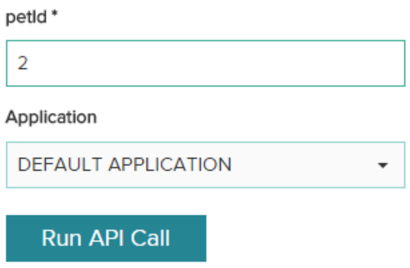
You can access the API at http://localhost:9100/petstoretest/1.0.0/pet/{petId}

Parameters

Name	Type	Description	Is required	Data Type
petId	path	ID of pet that ...	true	integer

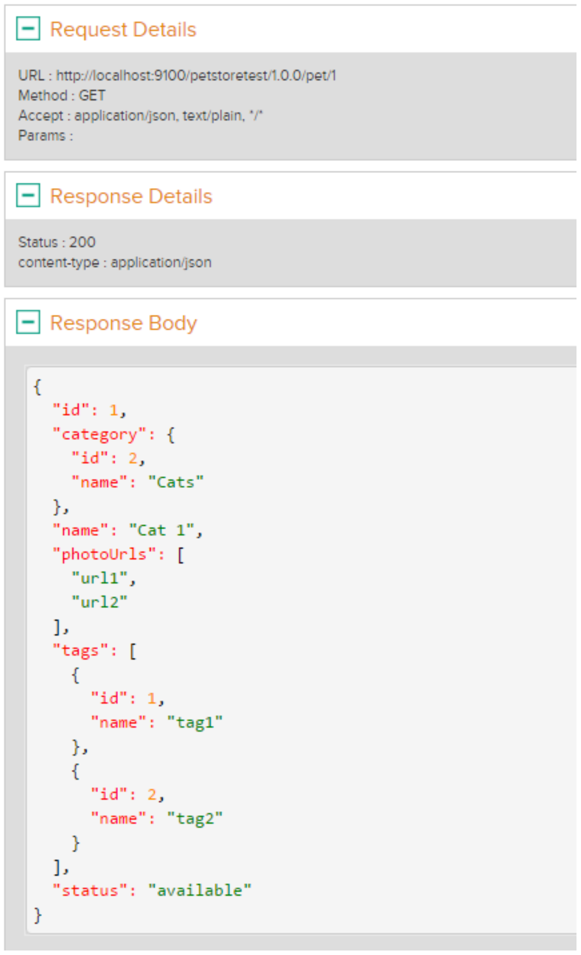
Authentication Type none

4 To test the API, click **Try Out**. Enter the value of petid and select an application. Click **Run API Call**.


petId * 2

Application DEFAULT APPLICATION

You can see the JSON-formatted response of the API call. For example,



URL : http://localhost:9100/petstoretest/1.0.0/pet/1
Method : GET
Accept : application/json, text/plain, */*
Params :

Status : 200
content-type : application/json

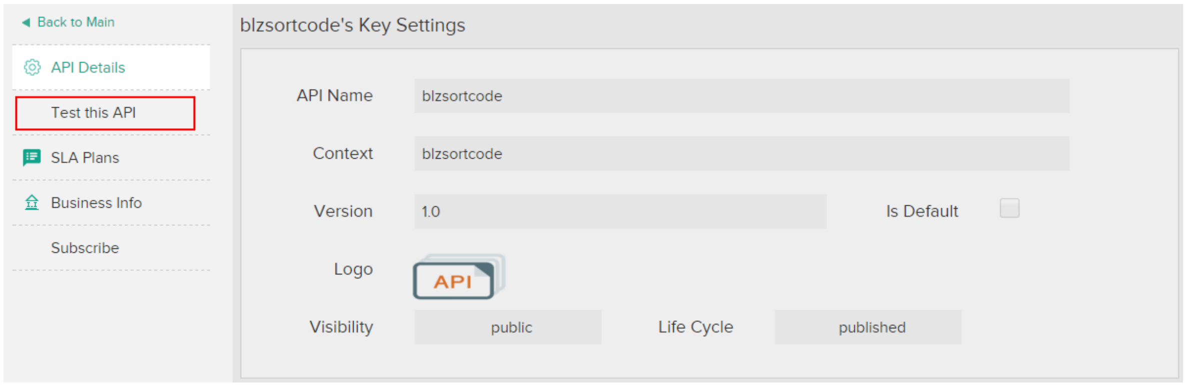
```
{
  "id": 1,
  "category": {
    "id": 2,
    "name": "Cats"
  },
  "name": "Cat 1",
  "photoUrls": [
    "url1",
    "url2"
  ],
  "tags": [
    {
      "id": 1,
      "name": "tag1"
    },
    {
      "id": 2,
      "name": "tag2"
    }
  ],
  "status": "available"
}
```

Consuming a SOAP API

Once an API publisher publishes a SOAP API, you can try the API and use the API in your applications. Go to the API catalog to view a list of all public APIs you can use.

1 Choose the API you want to tryout from the list of APIs.

2 Click **Test this API** from the left panel of the API details screen.


◂ Back to Main

API Details

Test this API

SLA Plans

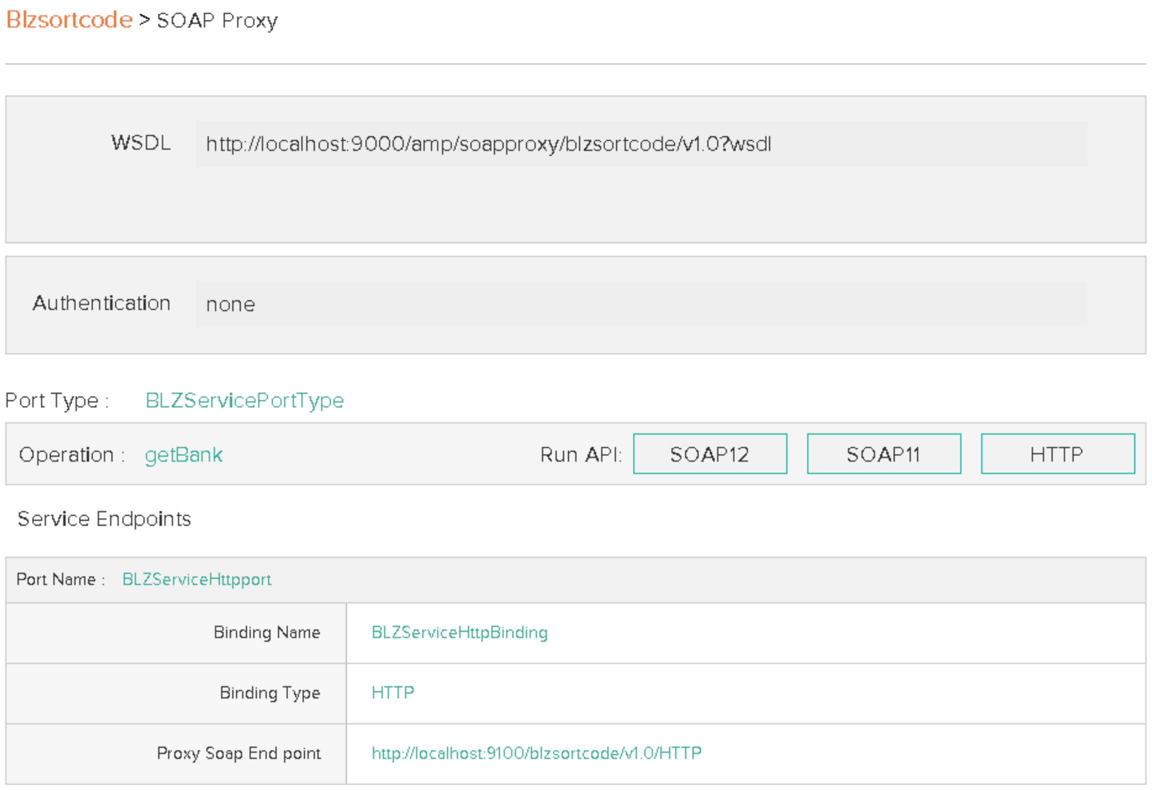
Business Info

Subscribe

blzsortcode's Key Settings

API Name	blzsortcode		
Context	blzsortcode		
Version	1.0	Is Default	
Logo	API		
Visibility	public	Life Cycle	published

3 You can see the proxy WSDL, WSDL port type, and the operation. You can also see the REST endpoints and their bindings. Choose the appropriate SOAP type.


Blzsortcode > SOAP Proxy

WSDL	http://localhost:9000/amp/soapproxy/blzsortcode/v1.0?wsdl
Authentication	none

Port Type : BLZServicePortType

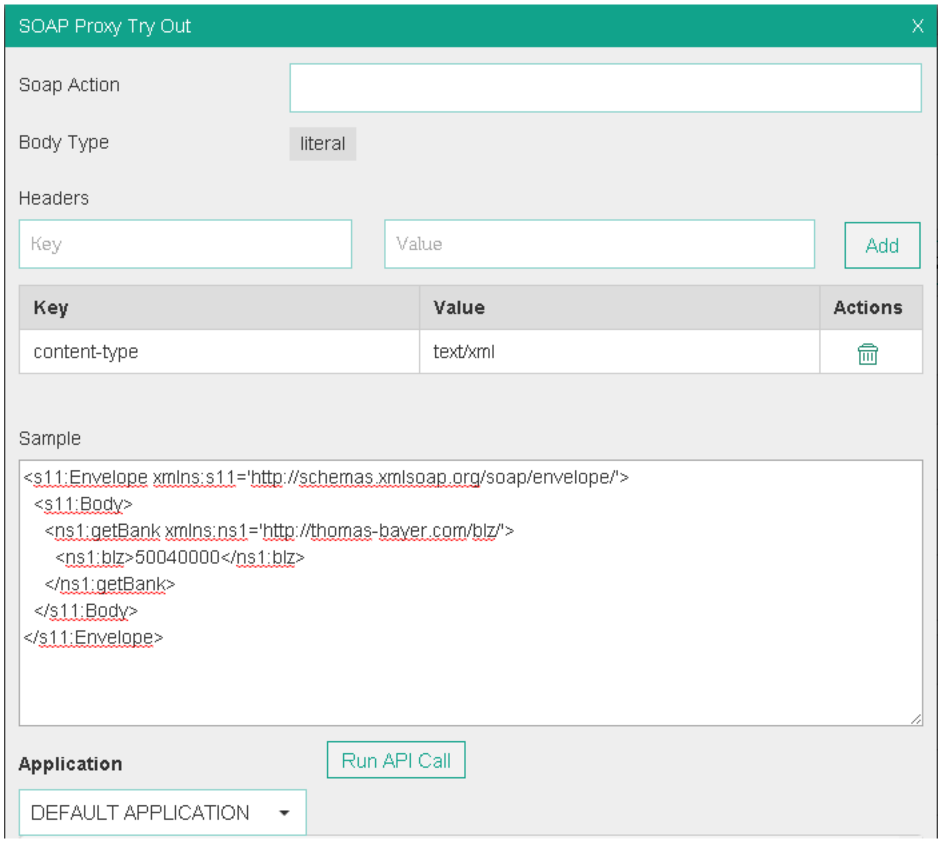
Operation : getBank   Run API: SOAP12 SOAP11 HTTP

Service Endpoints

Port Name : BLZServiceHttpport

Binding Name	BLZServiceHttpBinding
Binding Type	HTTP
Proxy Soap End point	http://localhost:9100/blzsortcode/v1.0/HTTP

4 Enter a BLZ code in the SOAP body and choose an application. For example, enter the BLZ code 50040000. Click **Run API Call**.



Soap Action

Body Type literal

Headers

Key	Value	Actions
content-type	text/xml	🗑

Sample

```
<s11:Envelope xmlns:s11='http://schemas.xmlsoap.org/soap/envelope/'>
 <s11:Body>
  <ns1:getBank xmlns:ns1='http://thomas-bayer.com/blz/'>
   <ns1:blz>50040000</ns1:blz>
  </ns1:getBank>
 </s11:Body>
</s11:Envelope>
```

Application

DEFAULT APPLICATION

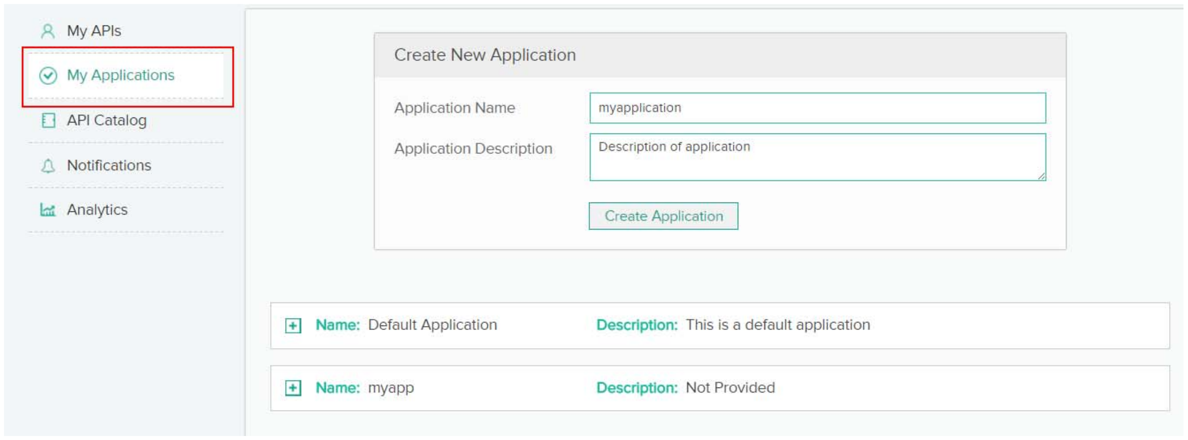
You can see the SOAP API returns the name of the bank that has the BLZ code 50040000. The response is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
       <ns1:getBankResponse
           xmlns:ns1="http://thomas-bayer.com/blz/">
           <ns1:details>
               <ns1:bezeichnung>Commerzbank</ns1:bezeichnung>
               <ns1:bic>COBADEFFXXX</ns1:bic>
               <ns1:ort>Frankfurt am Main</ns1:ort>
               <ns1:plz>60005</ns1:plz>
           </ns1:details>
       </ns1:getBankResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Creating an application

You can create an application and get your API subscribed to the application. Enter the name and description of the application you want to create and click **Create Application**.

You subscribe to a published API before using it in your applications. Subscription enables you to receive access tokens and be authenticated to call the API.



Application Name: myapplication

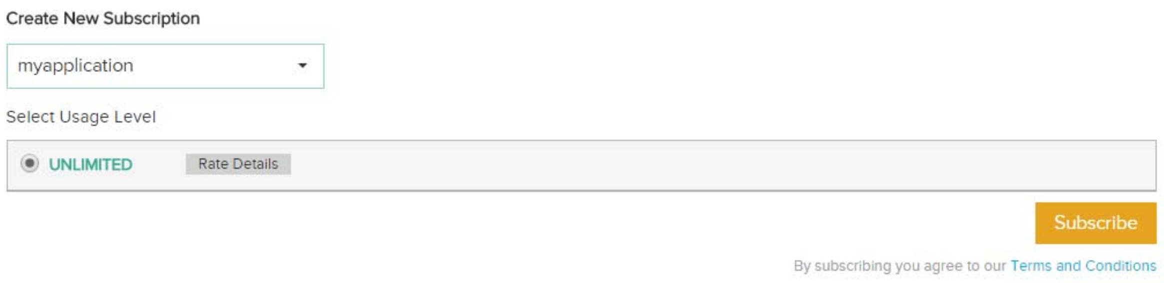
Application Description: Description of application

Name: Default Application Description: This is a default application

Name: myapp Description: Not Provided

Subscribing to an application

Once you create an application, you need an API to subscribe to the application. Choose the API and click **Subscribe** on the left pane. Select the application and choose the SLA plan. Click **Subscribe**. The publisher chooses the SLA plans for you.

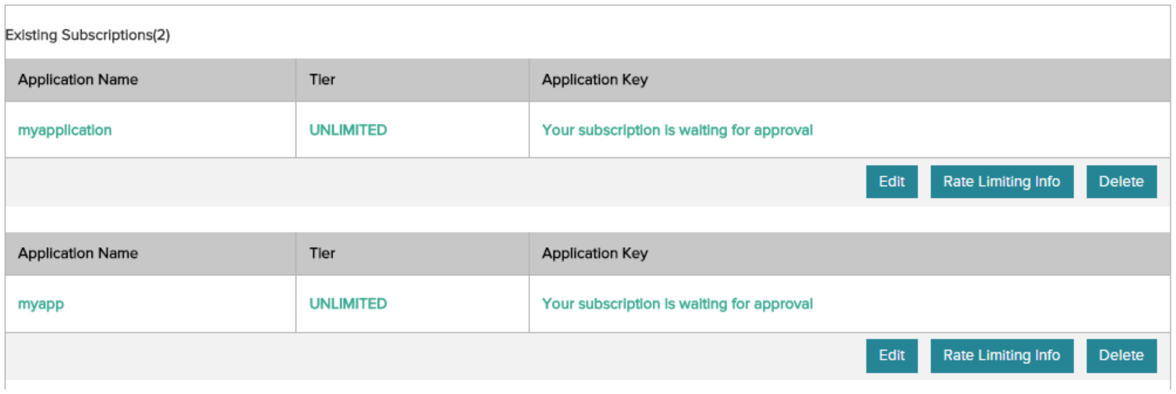


myapplication

● UNLIMITED

By subscribing you agree to our Terms and Conditions

You can see a list of applications that are waiting for approval from the API publisher. You can also modify and delete a subscription.



Existing Subscriptions(2)

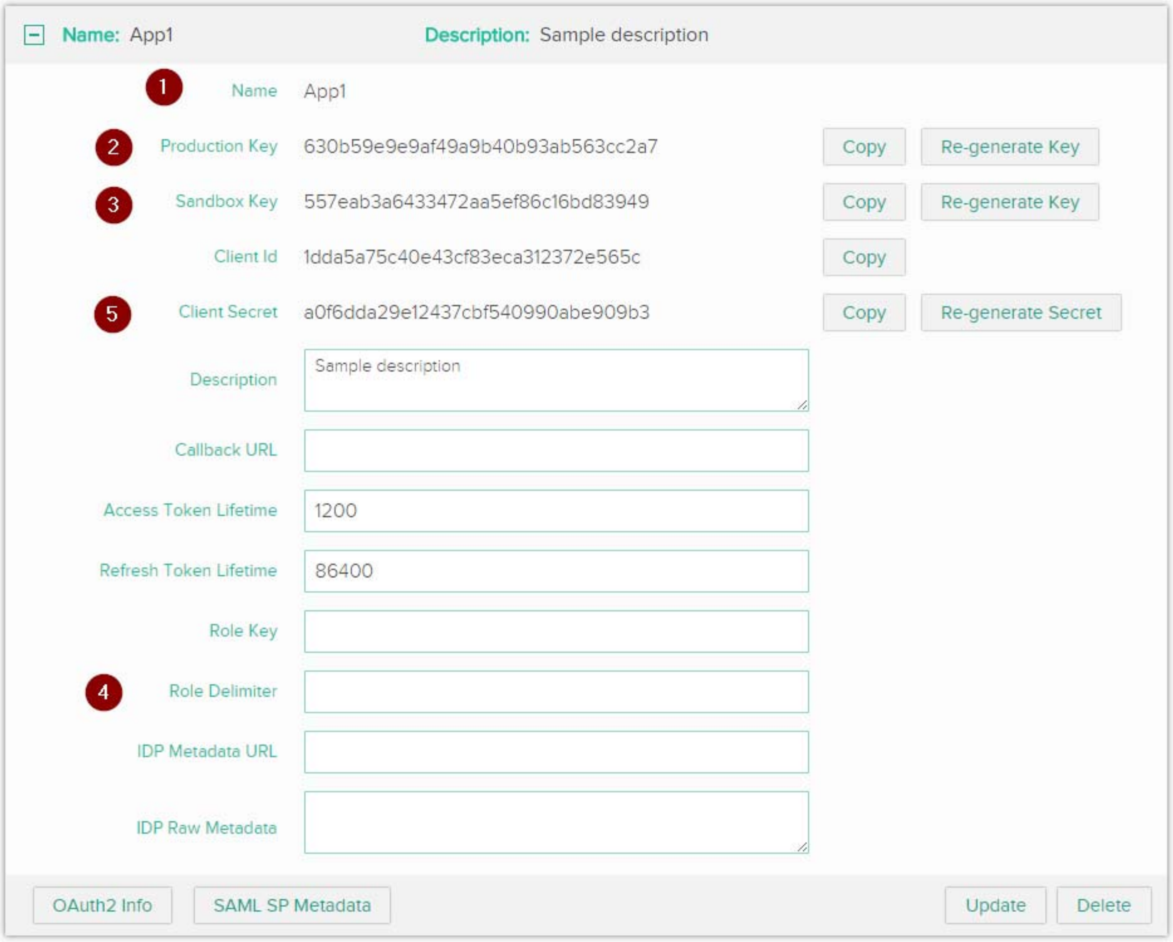
Application Name	Tier	Application Key
myapplication	UNLIMITED	Your subscription is waiting for approval
myapp	UNLIMITED	Your subscription is waiting for approval

Authorizing scope at application level

As a subscriber, you can select an application and restrict its scope.

1 To subscribe to an application, click Subscribe.

2 To see the security keys of the application, click Applications.



Name: App1 Description: Sample description

Name	App1
Production Key	630b59e9e9af49a9b40b93ab563cc2a7
Sandbox Key	557eab3a6433472aa5ef86c16bd83949
Client Id	1dda5a75c40e43cf83eca312372e565c
Client Secret	a0f6dda29e12437cbf540990abe909b3
Description	Sample description
Callback URL	
Access Token Lifetime	1200
Refresh Token Lifetime	86400
Role Key	
Role Delimiter	
IDP Metadata URL	
IDP Raw Metadata	

1	Name of the application.
2	This is an application key for a resource that has authentication of type apikey. When this key is used, the production endpoint of the API receives the request.
3	This is same as application key, but when this key is used, sandbox endpoint of the API receives the request.
4	The client ID uniquely identifies the application and is used when there is an authentication type of basic or Oauth2.
5	Client secret is used when authentication type is Oauth2 when obtaining the token using authorization code grant or client credentials flow. This must be kept secret.

Note: *Security Considerations- Items 2, 3, and 5 must be kept secret and not be exposed outside. If any of these get compromised, regenerate the keys so that the previous compromised keys get invalidated. Use the newly generated credentials in your application. When passing the credentials, use HTTPS.*

The Refresh Token Lifetime is always greater than the Access Token Lifetime.

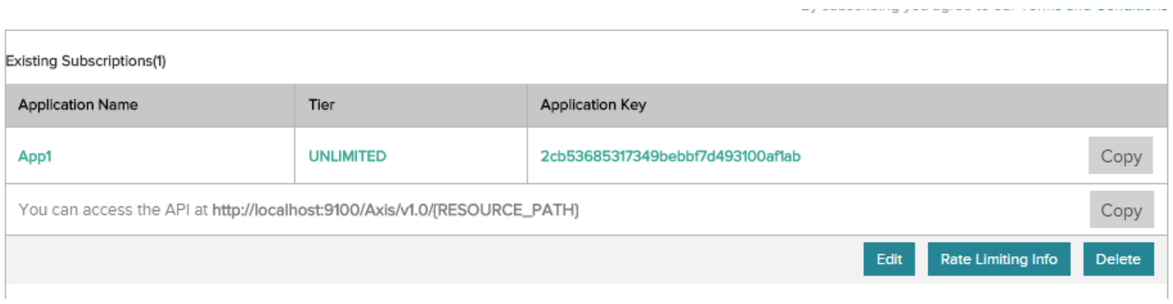
IDP Metadata URL: Specifies the metadata URL of the SAML identity provider. If any of the APIs subscribe using SAML and OAuth authentication, and your IDP hosts the metadata, enter the IDP URL here.

IDP Raw Metadata: Same as above. If you have the identity metadata, paste the XML here.

Only those tiers or usage plans associated with an approval goes to the publisher. The rest is auto-approved.

Authenticating API at subscription level

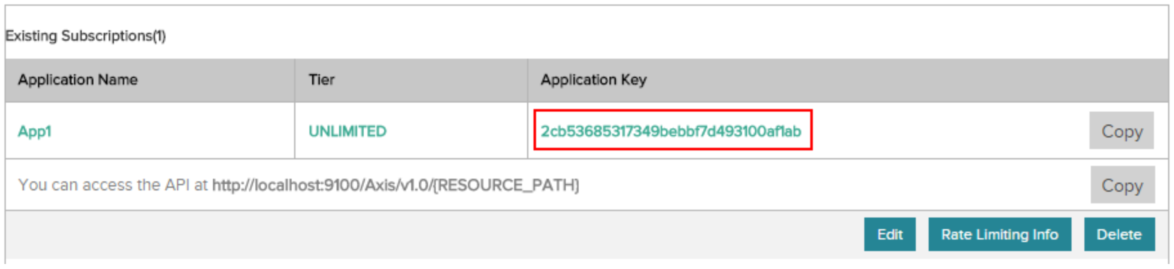
You can only make an API request once the administrator approves your subscription request. The following image represents the API subscription approval from the publisher.


Existing Subscriptions(1)

Application Name	Tier	Application Key
App1	UNLIMITED	2cb53685317349bebbf7d493100af1ab

You can access the API at http://localhost:9100/Axis/v1.0/{RESOURCE_PATH}

Once the publisher approves your application request, you can see the application key. You can use this key to make the request.

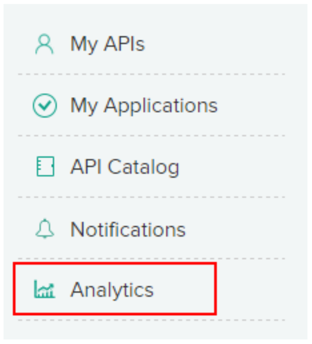

Existing Subscriptions(1)

Application Name	Tier	Application Key
App1	UNLIMITED	2cb53685317349bebbf7d493100af1ab

You can access the API at http://localhost:9100/Axis/v1.0/{RESOURCE_PATH}

Viewing the analytics dashboard

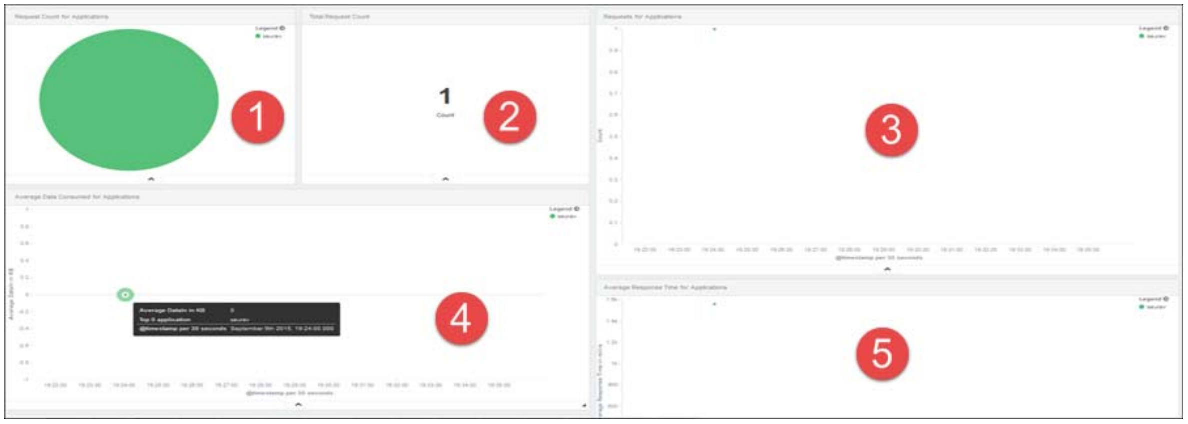
The analytics dashboard for a subscriber consists visualizations for the number of applications, number of API requests, and API errors. Click Analytics to launch the Subscriber analytics dashboard page.



There are three types of dashboards in the Subscriber portal:

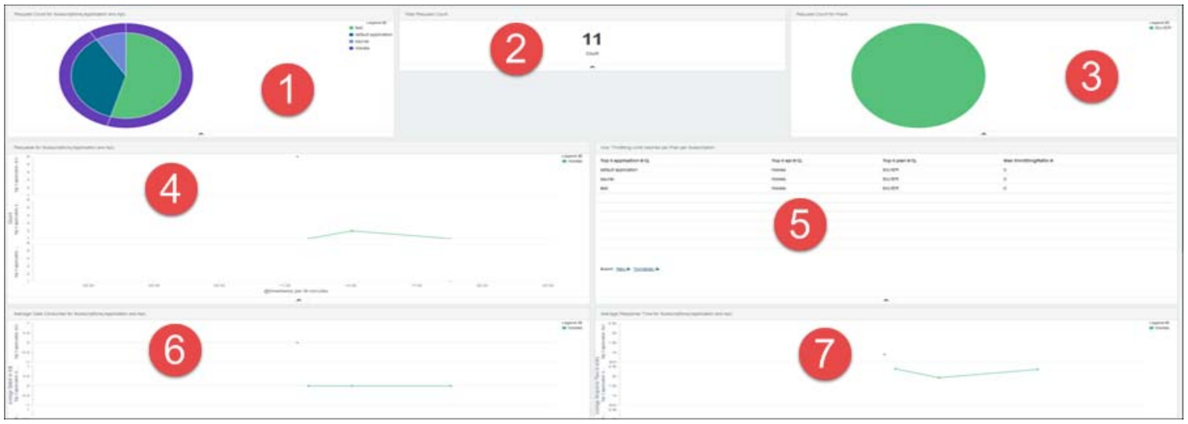
- Applications
- Subscriptions
- Errors

Applications: On the Applications panel, you can see the following visualizations:



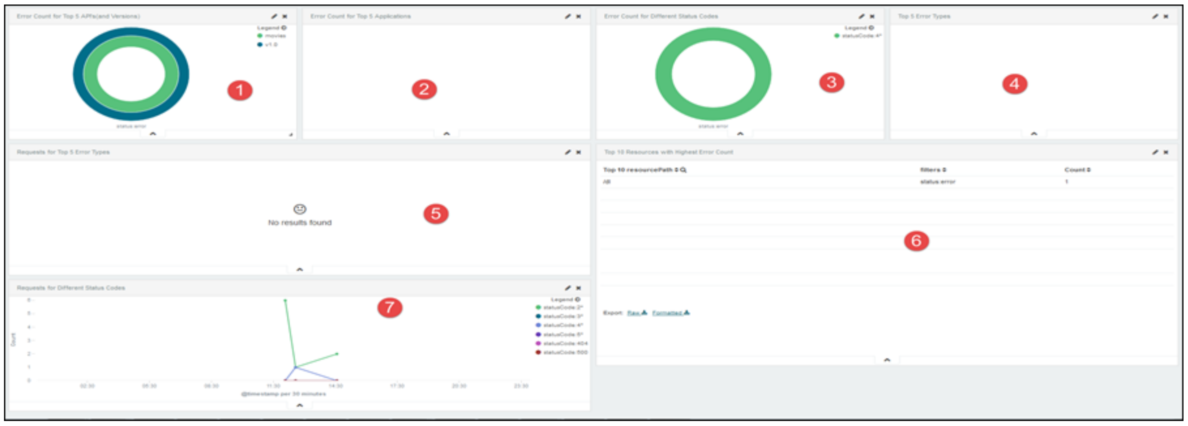
1	Pie-chart for the number of requests for applications.
2	The number of API requests.
3	Line-chart for the number of application requests.
4	Line-chart for the average data consumption by the applications.
5	Line-chart for average response times for applications.

Subscriptions: On the Subscriptions panel, you can see the following visualizations:



1	Pie-chart for the number of requests for subscriptions (applications and API).
2	The number of API requests.
3	Pie-chart for the number of requests for SLA plans.
4	Line-chart for the number of requests for subscriptions (applications and API).
5	List for maximum throttling limit per SLA plan.
6	Line-chart for average data consumption.
7	Line-chart for average response time for subscriptions.

Errors: On the Errors panel, you can see the following visualizations:



1	The number of errors for subscriptions and SLA plans.
2	The number of errors for different status codes.

3	Line-chart for requests for all status codes.
4	Pie-chart for the top five types of error.
5	List for top ten resources with maximum number of errors.

Filtering data according to time range

In the Analytics page, you can filter the results according to a time range.

There are three ways options:

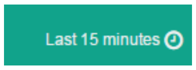
1 Quick

2 Relative

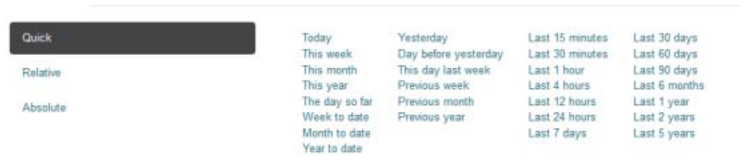
3 Absolute

To filter the data according to time range:

1 Click the time filter as shown below:

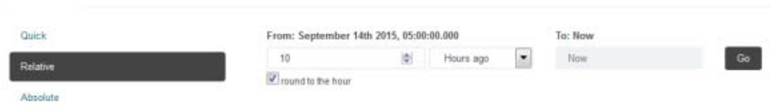


2 Select any time range from the list, as shown below:



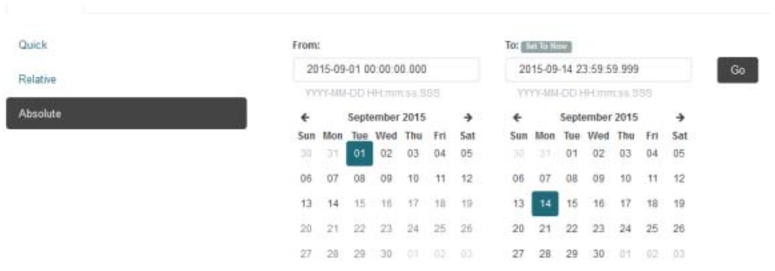
If you select Today, you can see the analytics of all APIs for the last 24 hours.

3 Click Relative. You can filter the data from a specified date and time to the current date and time.



From: September 14th 2015, 05:00:00.000
10 Hours ago
[✓] round to the hour
To: Now
Now

4 Click Absolute. You can filter the results according to dates.



From: 2015-09-01 00:00:00.000
YYYY-MM-DD HH:mm:ss.SSS
To: 2015-09-14 23:59:59.999
YYYY-MM-DD HH:mm:ss.SSS

Throttling and Rate-limiting

Concept Overview

Throttling

Throttling is a process that is used to control the usage of APIs by consumers during a given period. You can define throttling at the application level and API level. Throttling limit is considered as cumulative at API level.

Administrators and publishers of API manager can use throttling to limit the number of API requests per day/week/month. For example, you can limit the number of total API requests as 10000/day.

When a throttle limit is crossed, the server sends 429 message as HTTP status to the user with message content as "too many requests".

Throttling in API manager can be of two types: Hard and soft.

Hard: The number of API requests cannot exceed the throttle limit.

Soft: In this type, you can set the API request limit to exceed a certain percentage. For example, if you set the exceed limit to 90%, user gets a notification when the exceed limit is crossed.

Rate-limiting

Rate-limiting is a process that is used to define the rate at which consumers can access APIs. Also, it determines the speed at which a consumer can access APIs. Rate limit is calculated in real time.

Rate-limiting is applicable to resources. For example, in the following endpoint URL, "http://endpointurl/products", "/products" is a resource. It can be configured or over-ridden resource level.

You can add rate limits to API resources for the SLA plans to which the APIs are subscribed. Once the API exceeds the rate limit, the subscriber gets Status 429 message in the response header. Status 429 message indicates that the rate-limit has crossed.

For more information on adding rate limits to API resources, refer **Adding rate limits to API resources** section in Publisher document. Publisher assigns multiple plans for APIs while publishing them. Subscriber can select one of the plans while consuming APIs.

Administrators and publishers of API manager can use rate limiting to define the number of API requests per second/minute/hour. For example, if you set the rate as 5 req/sec, the speed at which a consumer can access each API can be high.

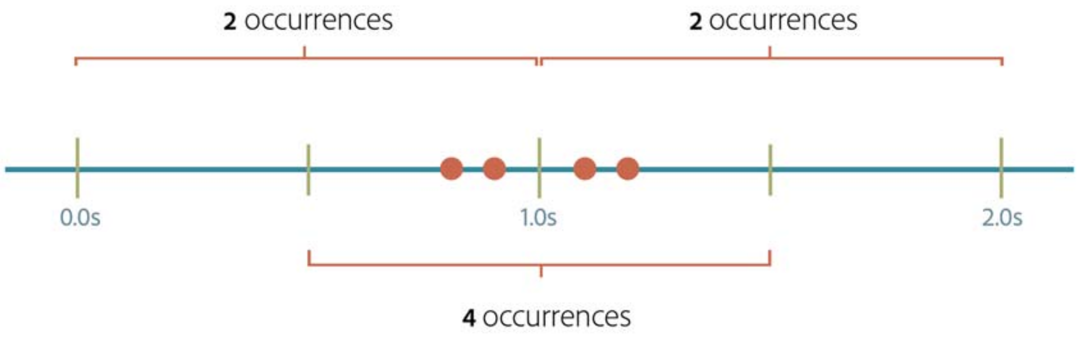
If API Manager is deployed in a cluster, the rate-limit is considered across the nodes. The consumer gets an error message whenever the defined limit is crossed within a single node or across a combination of nodes.

In API manager, two types of algorithms can be used to limit the rate: Rolling and Fixed.

In **fixed window algorithm**, the period is considered from the starting of the time unit to the end of the time unit. For example, a period is considered as 0-60 seconds for a minute irrespective of the time frame at which the API request has been made.

In **rolling window algorithm**, the period is considered from the fraction of the time at which the request has been made to the end of the time unit. For example, if two requests for API calls are made at 30th second and 40th second of a minute it is considered as two requests from 30th second of that minute up to the 30th second of next minute.

Consider an illustration as follows:



Let us assume there is a rate limit as 2 req/sec. In the above snapshot, you can notice two requests/occurrences before 1.0 second and two more occurrences after 1.0 second.

If you apply fixed window algorithm to this illustration, then two requests/occurrences are considered within 1 second. If you apply rolling window algorithm, then four occurrences are considered within 1 second.

SLA configuration

SLA configuration in API manager enforces access control to the users through rate-limiting and throttling. For instructions to configure SLA as an Administrator, refer to the Administrator document .

As a publisher, you can create a subscription plan and assign throttling limits to an API. For more information on specific instructions, refer to **Add tiers to an API** section of Publisher document .

Notifications

Overview

The API Manager sends notifications for any alerts or user events that require attention.

The are two ways through which a user can view the notifications:

1. When users log in to their respective portals, the notifications display on the portal
2. Users receive an email.

On the portal, the administrator, publisher, or the subscriber can view the notifications related to scenarios, such as:

- Request to approve API subscription
- Approved API subscription

- Termination or deprecation of an API
- Errors in API endpoints
- An SLA reaches its limits
- High CPU and memory usage
- System level messages

Notification service

The notification service in API Manager dispatches messages generated from various modules (security, tier, resources, and so on) to the intended recipients.

Alternatively, if the notifications are not configured, the notification service uses the default settings to send messages to various recipients.

Notification settings

You can configure the notification service at two levels:

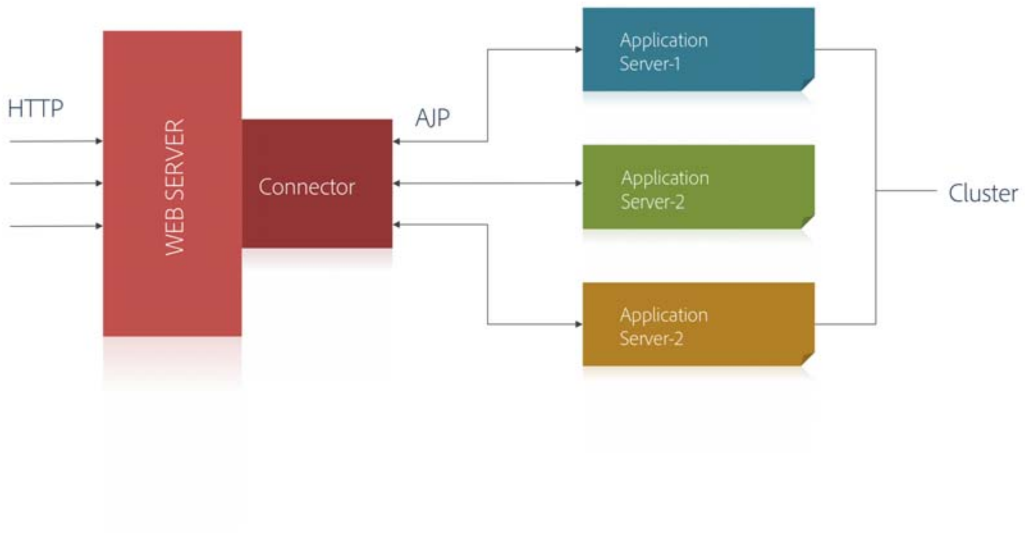
- **User level:** User level notification settings allow you to control the events and channels to receive notifications. The notifications can be edited in respective user portals.
- **System level:** System level settings allow you to control the deletion of notifications from a list. You can set a limit on the number of messages (for example, retain only the ten most recent messages).

Connectors

Overview

In the API Manager, connectors redirect the requests from web servers to application servers. Connectors also provide load balancing between various nodes in a cluster. Web server connectors in the API Manager manage the communication between the web server and the application server.

For Internet Information Services (IIS), the connector is isapi_redirect. On the other hand, for Apache web server, the connector is mod_jk.



In the API Manager, the connector communicates with a single cluster and a ColdFusion instance or cluster, if present.

To configure the same connector for API Manager and ColdFusion, make sure that the wsconfig folder of API Manager is present inside <cf_home>/<some-folder>. For example, C:\ColdFusion2016\apim\wsconfig.

Also copy the cluster.xml in the wsconfig folder file if you want to run wsconfig from outside the API Manager folder structure.

Connectors also refresh the information of a cluster dynamically. You need not restart the web server. There are some limitations:

- You cannot add a cluster.
- You can only add a node to a pre-configured cluster.
- You cannot modify the information of an existing node.

Configuration files

Connectors in the API Manager use the following configuration files:

- **worker.properties:** This file contains information about all application servers and clusters.
- **uriworkermap.properties:** This file contains rules of when to redirect a client request. This file also contains uri prefixes and filename extensions mapped to application servers or clusters defined in the file worker.properties.

Configuring connectors using Wsconfig tool

You can use the Wsconfig tool to configure connectors. The Wsconfig tool supports both a Graphical User Interface (GUI) and a Command Line Interface (CLI). You can configure the connectors in the API Manager and ColdFusion, if installed.

The Wsconfig tool reads the configuration files and configures the connectors based on some parameters. You can use this tool to add or remove a connector. This tool can be run from any server, but it requires a configuration file to get the information about a cluster and its nodes.

You can refresh information in a cluster after adding a node. Enter wsconfig -help in the command prompt and view the parameters for refresh.

Note: *To serve PUT and DELETE requests through connector in IIS, remove the following:*

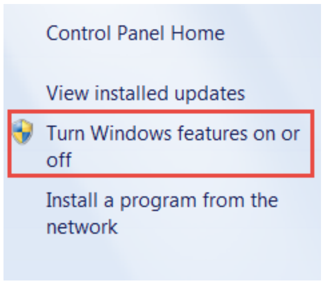
1 webDAVModule from IIS > Modules

2 WebDAV from IIS > Handler Mapping in IIS Manager

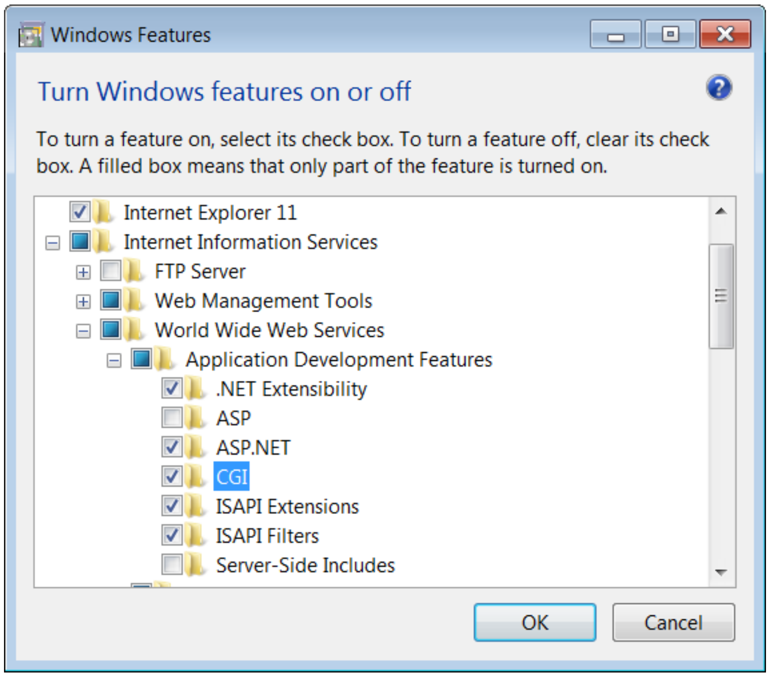
Configuring IIS

1 Click Start > Control Panel. Select Programs and Features.

2 Click Turn Windows features on or off.


Control Panel Home
View installed updates
Turn Windows features on or off
Install a program from the network

3 Select Internet Information Services > World Wide Web Services > Application Development Features.



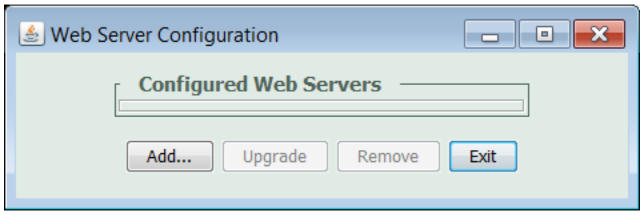
To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- [✓] Internet Explorer 11
- [■] Internet Information Services
  - [ ] FTP Server
  - [■] Web Management Tools
  - [■] World Wide Web Services
    - [■] Application Development Features
      - [✓] .NET Extensibility
      - [ ] ASP
      - [✓] ASP.NET
      - [✓] CGI
      - [✓] ISAPI Extensions
      - [✓] ISAPI Filters
      - [ ] Server-Side Includes

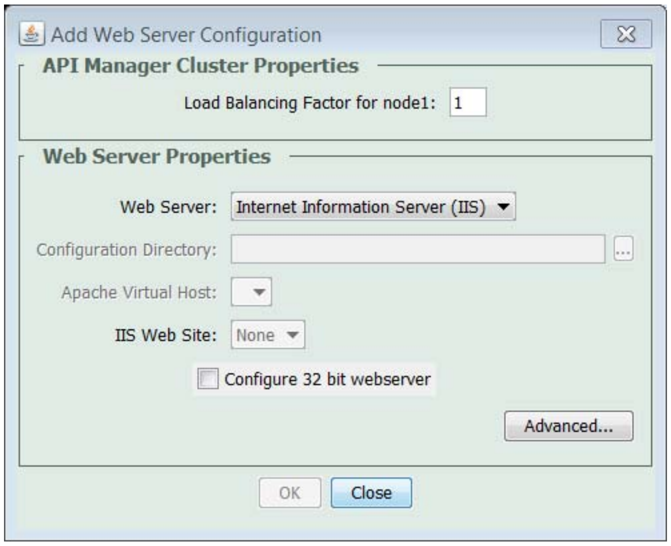
Select the following check-boxes:

- ASP.NET
- CGI

4 In the <API Manager Installation Directory>/wsconfig/ directory, double-click wsconfig.exe.



5 Click Add.



Load Balancing Factor for node1: 1

Web Server: Internet Information Server (IIS)

Configuration Directory:

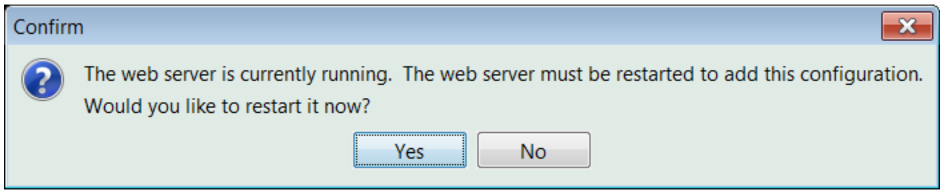
Apache Virtual Host:

IIS Web Site: None

Configure 32 bit webserver

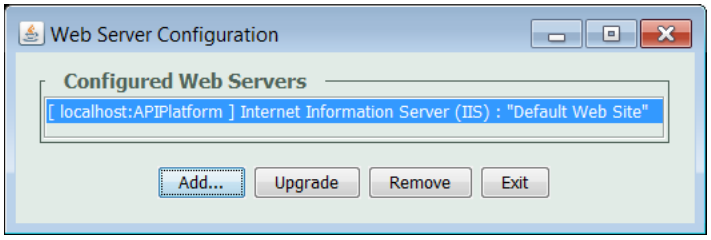
6 From the IIS Web Site drop-down list, select Default Web Site. Click OK.

7 Restart the server. Click Yes.



The web server is currently running. The web server must be restarted to add this configuration. Would you like to restart it now?

The default IIS server becomes the default web site.



[localhost:APIPlatform] Internet Information Server (IIS) : "Default Web Site"

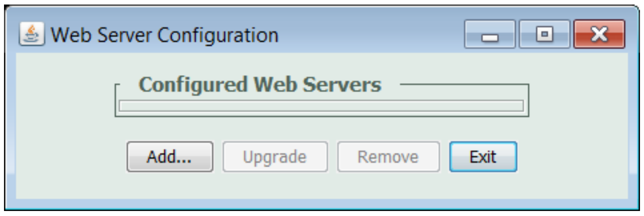
8 To verify the successful installation of the connector, enter <localhost>:80/portal in the address bar of a web browser.

__Note:__ The Administrator is blocked in the web server. To access the portal, use the port, 9000.

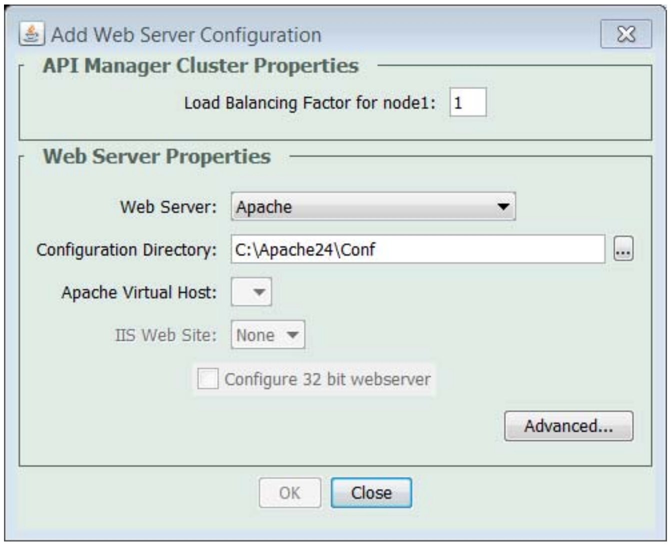
To configure a web server on a remote machine, copy the cluster.xml file to the folder <API Manager installation directory>/conf/. Start the API Manager first and then run the wsconfig tool.

Configuring Apache

1 In the <API Manager Installation Directory>/wsconfig/ directory, double-click wsconfig.exe.



2 Click Add. On the Add Web Server Configuration dialog box, select Apache from the Web Server drop-down list.



Load Balancing Factor for node1: 1

Web Server: Apache

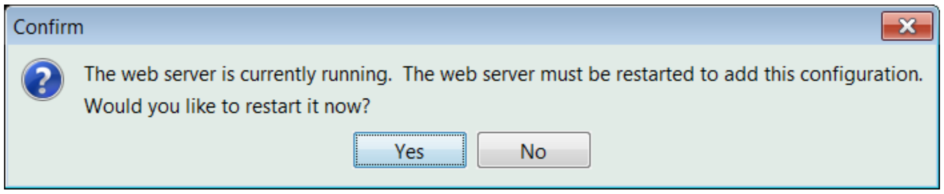
Configuration Directory: C:\Apache24\Conf

Apache Virtual Host:

IIS Web Site: None

Configure 32 bit webserver

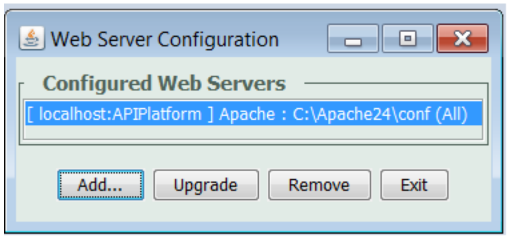
3 Restart the server. Click Yes.



Confirm

The web server is currently running. The web server must be restarted to add this configuration. Would you like to restart it now?

The Apache server becomes the default web server.



Web Server Configuration

Configured Web Servers

[localhost:APIPlatform] Apache : C:\Apache24\conf (All)

You can configure connectors for a particular Apache Virtual Host, similar to IIS. The Virtual Host serves requests only to the API Manager, not to any non-API Manager configurations.

Note: *If you use the command line, specify -vhost <virtual_host_name>. The parameter is mandatory.If you do not have a virtual host or you want to install Apache globally, use -vhost All in the command line.*

Setting up Cluster Support

Overview

A cluster consists of a set of objects that act as a single instance, and divide the workload. Clustering improves performance as the requests are distributed across several objects. Clustering also improves reliability because if one object instance is unavailable or experiencing high traffic, other instances can handle the request.

The API Manager provides cluster support for application scaling and high availability. As an administrator, you can create a homogeneous cluster for scaling and availability for your APIs via the API Manager.

Starting a cluster

To start a cluster in the API manager:

? Change <cluster><enabled>false</enabled></cluster> to <cluster><enabled>true</enabled></cluster>.

To add a node to the cluster, start the node on a separate port. Any change in the API information in one node reflects in all nodes within a cluster.

If you add a new node to the cluster, perform the following SAML-related tasks:

1. Copy the file seed.properties to the new node. You can locate the file in {API Manager_HOME}/conf/.
2. Copy the file samlspcerts.jks to the new node. You can locate the file in {API Manager_HOME}/conf.
3. Copy the following snippet in the config.xml file to the new node. You can locate the file in {API Manager_HOME}/config.

```
<keystore>
   <path>C:\ColdFusion2016APIManager\conf\samlspcerts.jks</path>
   <type>JKS</type>
   <password>eaf7e40627b249f7ab09d01b01efde77</password>
   <samlspalias>ampserviceprovider</samlspalias>
<samlspkeypassword>8d4ba4d9cad64d31a29e6b05221e103a</samlspkeypassword>
</keystore>
```

Using the API Manager data store for clustering

When installing API Manager, you can opt to install the API Manager data store. All API Manager nodes that are a part of a cluster share a common data store. This server stores all configuration-related information of the API Manager. All nodes connected to the server share similar configuration.

Each server in the API Manager has its own cluster configuration. This configuration retrieves the connection details from the data store when a node starts up. In a clustered environment, the API Manager node connects to the server defined in the cluster configuration.

Accessing a node in Redis

By default when Redis is installed, it is reachable only through localhost (127.0.0.1). To access it from a remote machine, perform the following steps:

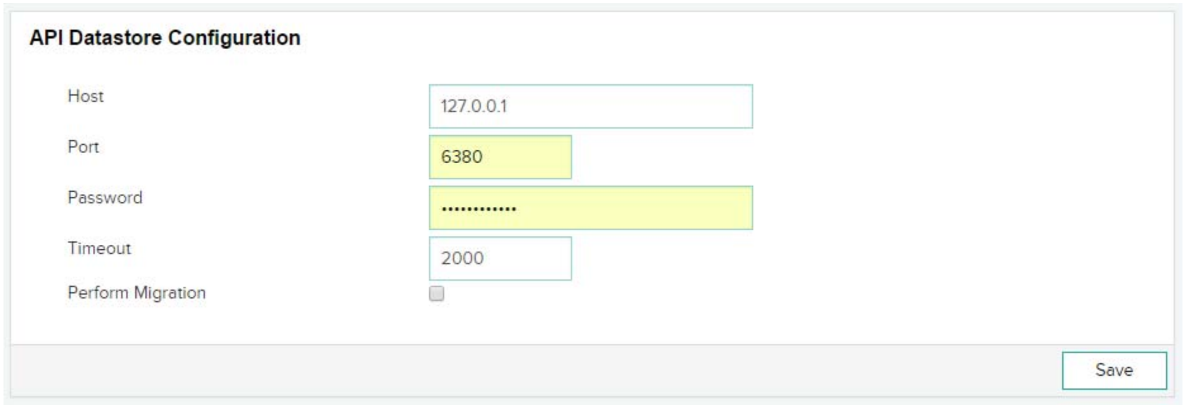
1 Navigate to <API Manager installation directory>\database\datastore and open the file redis.windows.conf.properties. In case of Linux, it is redis.conf.properties.

2 Specify the host name or IP address of the Redis data store in the bind section so that it can be accessible to remote API Manager nodes. You also perform this step when installing an API Manager node on a machine other than Redis datastore machine.

Viewing the cluster via the portal

You can view the cluster via the Administration portal if you install the API Manager stand-alone in your computer.

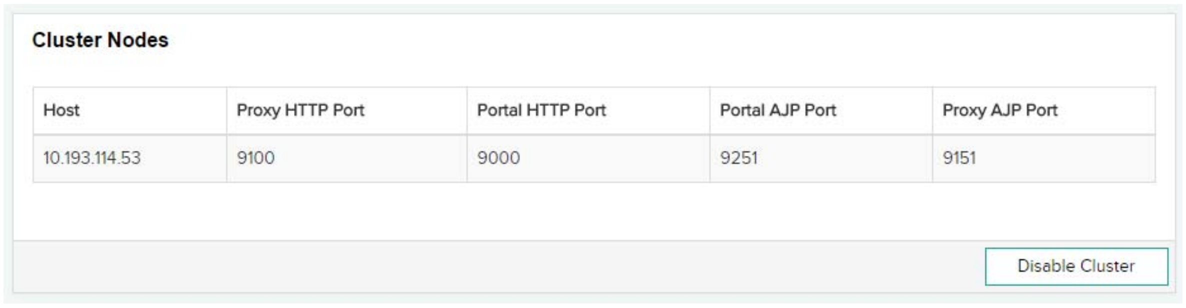
On the Admin portal, you can view the details of the data store to connect to.



API Datastore Configuration

Host	127.0.0.1
Port	6380
Password	••••••••••••
Timeout	2000
Perform Migration	☐

On the Cluster Configuration page in the Administrator portal, you can view the list of nodes that are a part of a cluster. This information persists as part of the node configuration.



Cluster Nodes

Host	Proxy HTTP Port	Portal HTTP Port	Portal AJP Port	Proxy AJP Port
10.193.114.53	9100	9000	9251	9151

Accessing a node in Elastic Search

To access Elastic Search from a remote machine, perform the following steps:

1 Navigate to <API Manager installation directory>\database\analytics\config and open the file elasticsearch.yml.

2 Specify the host name or the IP address of the Elastic Search server in the network.host section so that the address is accessible to remote API Manager nodes.

Fault tolerance through clustering

Since clustering in the API Manager is homogenous, APIs published from one node are published from all nodes in the cluster. If any node fails, there is no impact on availability of the published APIs.

ColdFusion and API Manager - Integration

Overview

API Manager facilitates the management of APIs that are exposed through REST or SOAP services. To experience seamless integration with ColdFusion APIs, you can access ColdFusion REST services from API Manager.

The admin portal available with the API Manager lets you define the ColdFusion discovery server. Once such a ColdFusion discovery server is set, you can readily import REST services defined in the ColdFusion server with just a single click. With minimal configuration, you can readily publish the APIs associated.

Starting API Manager with ColdFusion

You can start API Manager from ColdFusion in either of the following ways:

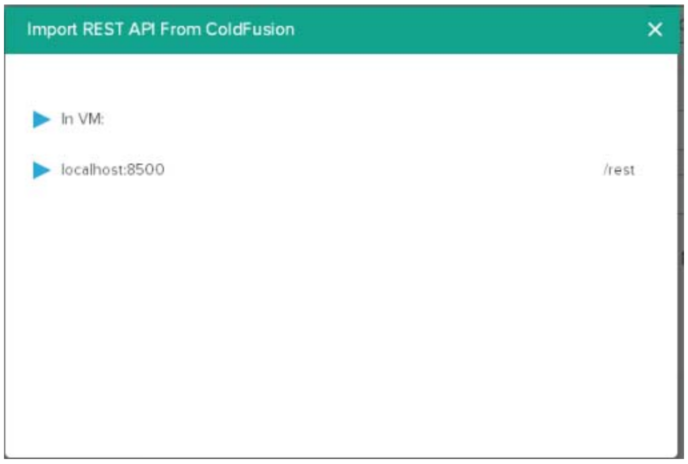
- Provide ColdFusion's home directory while installing API Manager.. API Manager is installed in a separate directory.
- Provide the –Dapim.home={application-home} flag in <ColdFusion Home>.

Discover ColdFusion REST services from API Manager

InVM

If you start API Manager along with ColdFusion, the API Manager detects REST services that are published in ColdFusion.

Publisher can import these REST services from the “In VM” category in the Import screen as shown below:



Note: ***This category appears only when API Manager is started with ColdFusion.***

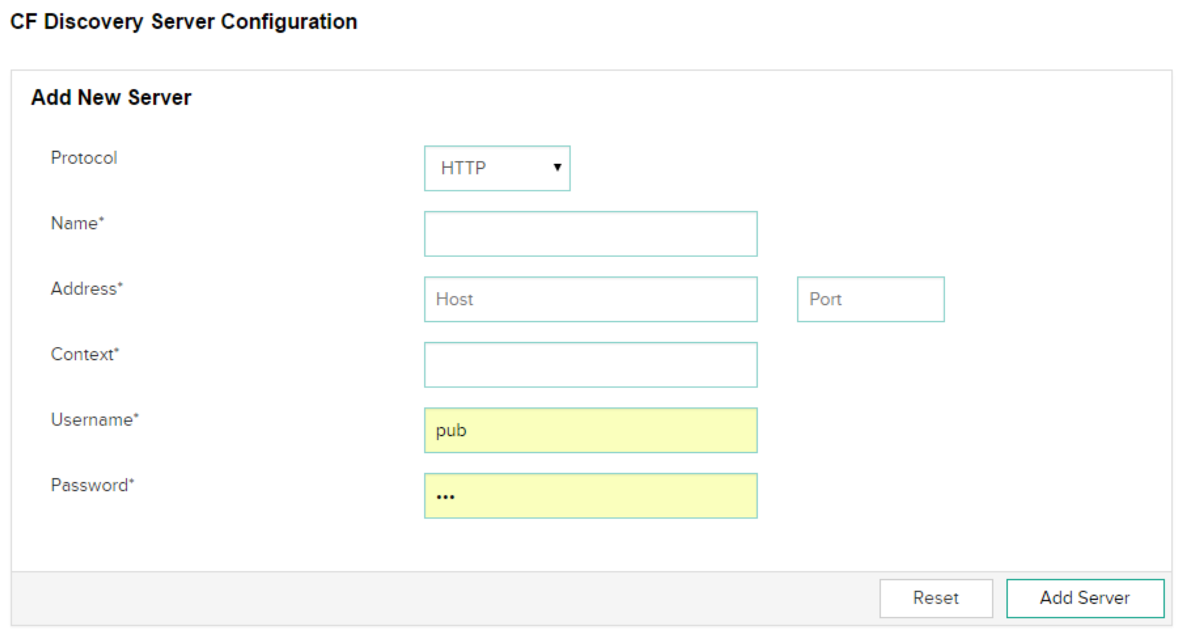
Remote

If you have already installed API Manager and you want to discover REST services published in a non-InVM ColdFusion, add the ColdFusion server configuration according to the following steps:

1 Start API Manager in Administrator mode.

2 Click **CF Discovery Server Configuration** on the left pane.

3 Add the configuration values.

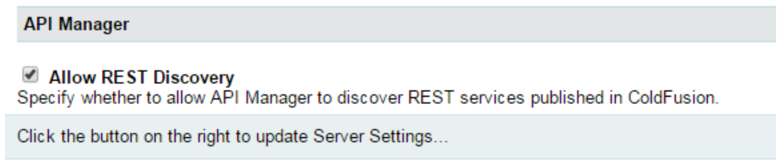

CF Discovery Server Configuration

Add New Server

Protocol	HTTP ▾	
Name*		
Address*	Host	Port
Context*		
Username*	pub	
Password*	•••	

Set API Manager flags in ColdFusion

From ColdFusion Administrator > Settings page, you can set the flag for REST discovery service as shown in the following screenshot.


API Manager

✓ Allow REST Discovery
Specify whether to allow API Manager to discover REST services published in ColdFusion.

Click the button on the right to update Server Settings...

If you select the **Allow REST Discovery** check box option, then API Manager discovers REST services automatically. If you clear this option, then REST services do not appear in API Manager's Publisher page while importing APIs.

Note: *If you clear this check box during CF startup and enable later, you can perform one of the following:*

- Restart the ColdFusion server.
- Reinitialize the REST service to make the service available in API Manager.

API Manager-Metrics and Logging

Overview

ColdFusion API Manager collects and analyzes information that APIs generate. This information is useful to measure the performance of APIs. The information provided by the API Manager can be used to make decisions, such as:

- Change in any API.

- Applications that are consuming an API for the maximum or minimum time.
- HTTP methods (GET, PUT, POST, and so on) that have maximum usage.
- Capacity planning of APIs.

Metrics

When an API makes a runtime request to the API Manager, the analytic server pulls the following information from the request, such as:

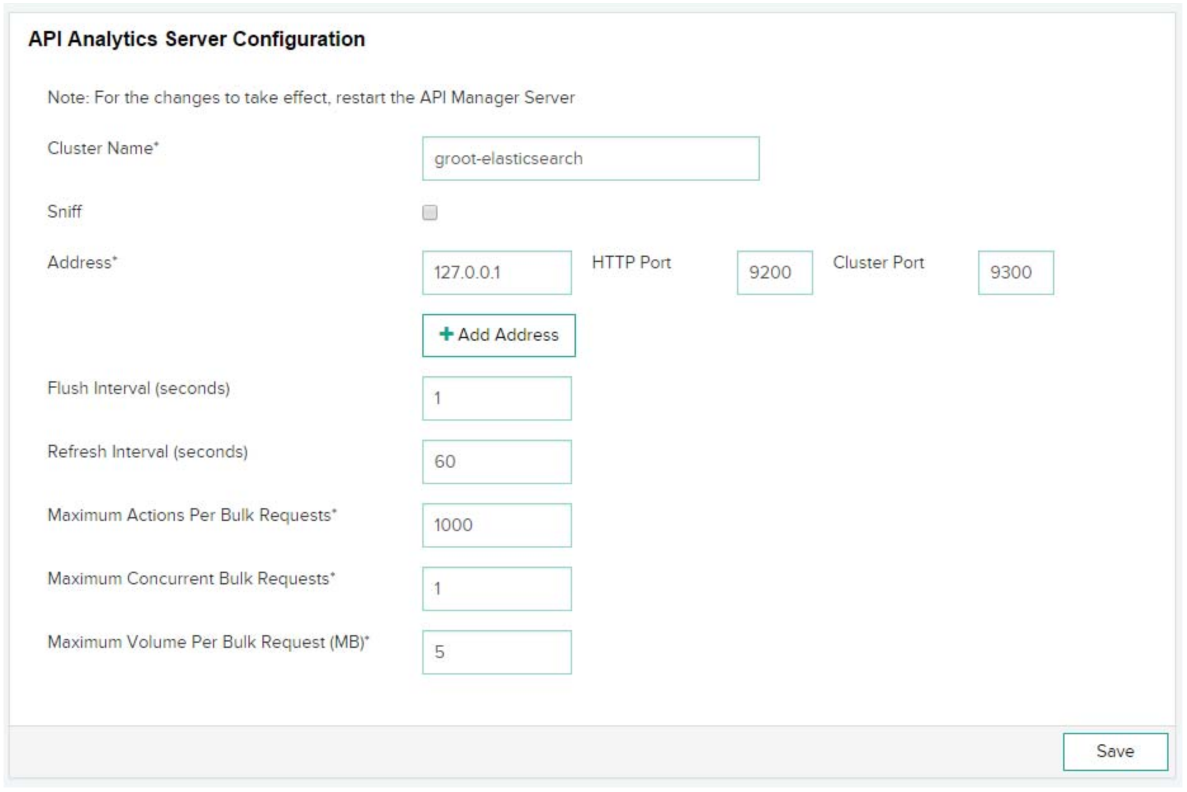
- Timestamp
- API Name
- Version
- Publisher
- Consumer
- Application
- SLA
- Error
- Request Method
- Status Code
- Request Content Size
- Response Content Size
- Request Time
- Sub resource pattern
- Request Content-Type
- Response Content-Type
- Cluster Node Name
- Request IP

To view the analytics, a publisher, consumer, or an administrator has access to a dashboard. Publishers and administrators have their own personal dashboards which they can edit and add a visualization, dashboard, and so on. But dashboards for consumers cannot be personalized.

This dashboard is a web interface that publishes metrics analytics and log information in the form of graphs and pie charts.

Modify metric settings

As an administrator, you can configure the settings for the number of API requests for which to generate metrics. On the Administrator portal, select Server > API Analytics Server Configuration to display the Metric Settings page.


API Analytics Server Configuration

Note: For the changes to take effect, restart the API Manager Server

Cluster Name*	groot-elasticsearch				
Sniff	☐				
Address*	127.0.0.1	HTTP Port	9200	Cluster Port	9300
Flush Interval (seconds)	1				
Refresh Interval (seconds)	60				
Maximum Actions Per Bulk Requests*	1000				
Maximum Concurrent Bulk Requests*	1				
Maximum Volume Per Bulk Request (MB)*	5				

Cluster Name- Enter the name of the cluster in which the APIs make the request. The analytics of these requests are then published into a portal. For more details, refer to Setting up Cluster Support.

Sniff-In Analytics Server configuration, the client can sniff the rest of the cluster, which adds nodes that the cluster can use. To enable the feature, select **Sniff**.

Flush Interval- Enter the time interval, in seconds, after which the ElasticSearch server receives a specified number of API requests.

Maximum Actions Per Bulk Request- Enter the number of API requests that are collectively sent to the ElasticSearch server for metrics calculation and visualization. If you enter 1000 in the field, no more than 1000 requests can be sent to the server.

Maximum Concurrent Bulk Request- Enter the number of concurrent blocks of API requests that are sent to the ElasticSearch server. For example, if you enter 2 in the field, no more than two concurrent blocks of API requests can be sent to the server.

Maximum Volume Per Bulk Request- Enter the size of the concurrent requests in this field. The size is calculated in MBs. For example, if you enter 5 in the field, the size of concurrent API requests cannot exceed 5 MB.

Log analysis in API Manager

A user (publisher or administrator or subscriber) can view the information that the API Manager logs.

Since the API Manager is a multi-tenant system, multiple publishers can use platform concurrently. So when a publisher retrieves the logged information, the API Manager retrieves those logs that are specific to a publisher. It is more secure since first publisher has no access to the second publisher's logs.

On the dashboard, the publisher can view and analyze logs using certain filters, such as:

- Logs in last 15 mins/30 mins/last day/last month/any custom time
- Error logs
- Logs from a specific consumer
- Logs for a particular service or a version
- A combination of one or more filters

Generating Swagger documents

Overview

Swagger is a project specification that is used to describe and document RESTful APIs. In Adobe ColdFusion (2016 release), you can create swagger doc automatically from REST CFC after it is implemented and registered in server. The Swagger version that is supported in ColdFusion is 1.2.

For more information on Swagger project overview, refer to Swagger documentation. Swagger specification 1.2 is available here.

Document generation process

The Swagger doc generation feature is a part of ColdFusion Server. ColdFusion server generates the Swagger doc automatically once you register REST CFC application.

Create your REST CFC file

You can create REST CFC application file of your choice and place this file in the root folder (wwwroot) of ColdFusion server. A sample CFC file content structure is shown in the following studentservice.cfc file.

```
<cfcomponent rest="true" restpath="studentService">

   <cffunction name="addStudent" access="remote" returntype="void" httpmethod="PUT"
description="add student">
       <cfargument name="name" type="string" required="yes" restargsource="Form"/>
       <cfargument name="age" type="numeric" required="yes" restargsource="Form"/>
           <!--- Adding the student to data base. --->
   </cffunction>
   <cffunction name="addStudents" access="remote" returntype="void" httpmethod="POST"
description="add students">
       <cfargument name="name" type="student[]" required="yes" restargsource="body"/>
           <!--- Adding the student to data base. --->
   </cffunction>
<cffunction name="deleteStudent1" access="remote" returntype="void" httpmethod="DELETE"
description="delete students">
       <cfargument name="students" type="student[]" required="yes" restargsource="Body"/>
           <!--- Adding the student to data base. --->
   </cffunction>
   <cffunction name="updateStudentAddress" access="remote" returntype="address"
httpmethod="POST" restpath="{studentId}" description="modify student address" hint="modify the
address for given studentId">
       <cfargument name="studentId" type="numeric" required="yes" restargsource="PATH" />
       <cfargument name="address" type="address" required="yes" restargsource="Body" >
           <!--- Adding the student to data base. --->
   </cffunction>
   <cffunction name="getStudent" access="remote" returntype="Student" restpath="{name}-{age}"
httpmethod="GET" description="retrieve student" produces="application/json"
responseMessages="404:Not found,200:Successfull:student" >
       <cfargument name="name" type="string" required="yes" restargsource="Path"/>
       <cfargument name="age" type="string" required="yes" restargsource="Path"/>
           <!--- Create a student object and return the object. This object will handle the
request now. --->
       <cfset myobj = CreateObject("component", "Student")>
       <cfset myobj.name = name>
       <cfset myobj.age = age>
       <cfreturn myobj>
   </cffunction>
</cfcomponent>
```

Using application.cfc file

If you do not want the ColdFusion server to generate swagger doc automatically, set the following code to false in application.cfc file. Refresh the registered application in CF administrator.

<cfset this.restsettings.generateRestDoc="false">

A sample application.cfc file is shown below for your reference.

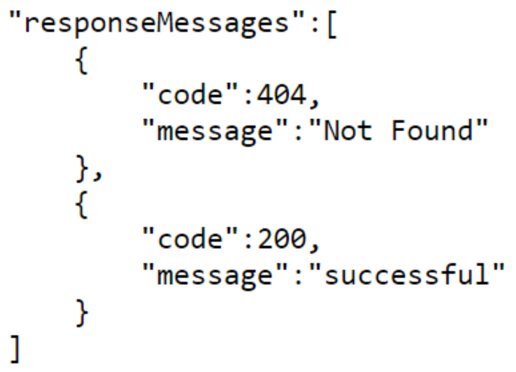
```
<cfcomponent>
<cfset this.name="info" />
<cfset this.sessionmanagement = true />
<cfset this.restsettings.generateRestDoc="true">
<cfset this.restsettings.restDocInfo.title="this is title">
<cfset this.restsettings.restDocInfo.apiVersion="2.0">
<cfset this.restsettings.restDocInfo.description="this is description">
<cfset this.restsettings.restDocInfo.termOfServiceUrl="url is here">
<cfset this.restsettings.restDocInfo.contact="xyz@adobe.com">
<cfset this.restsettings.restDocInfo.license="adobe 1.0">
       <cfset this.restsettings.restDocInfo.licenseUrl="http://abc.com">
</cfcomponent>
```

Using new response messages attribute

A new attribute named responseMessages has been introduced in Adobe ColdFusion (2016 release). You can use this attribute in REST CFC file as shown in the sample file below.

```
<cfcomponent rest="true" restPath="/cookieService" produces="text/plain" >
<!--- Test with various produces --->
<cffunction name="sayPlainHelloUser" responseMessages="404:Not
Found,200:successful,10:notdefine" access="remote" returnType="String" httpMethod="GET"
produces="text/plain">
    <cfargument name="nAme" type="string" restargsource="cOOkie" required="false"
default="CF">
    <cfset res="Hello " &amp; name>
    <cfreturn res>
</cffunction>
</cfcomponent>
```

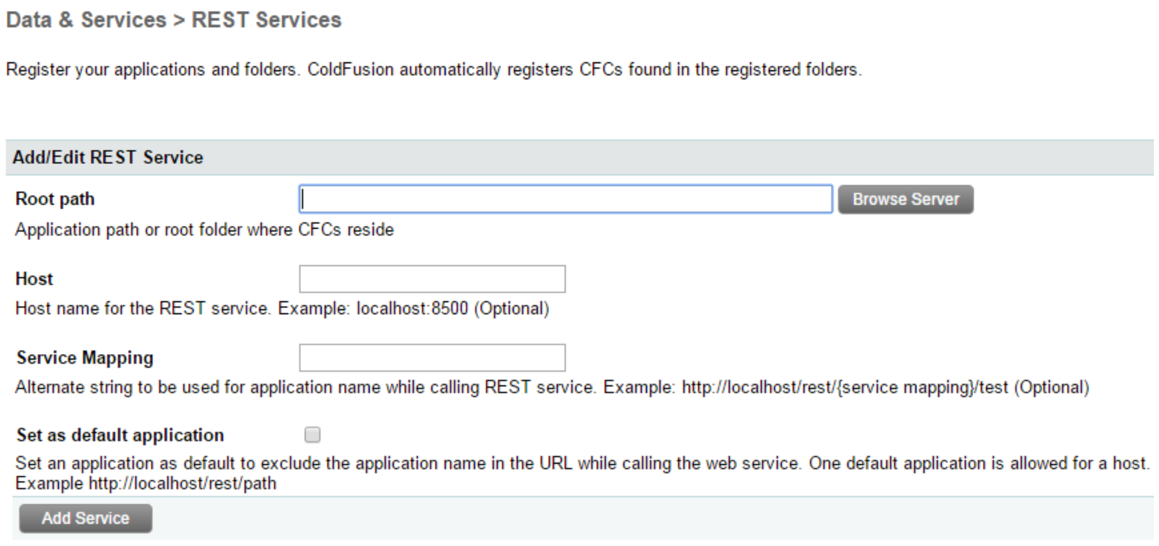
The swagger API document generated from this sample responseMessages code appears as shown below.



```
"responseMessages":[
     {
          "code":404,
          "message":"Not Found"
     },
     {
          "code":200,
          "message":"successful"
     }
]
```

Register your CFC application

Start ColdFusion Administrator. Click Data & Services > REST Services on the left pane and add configuration values as per the instructions in the following dialog.



Data & Services > REST Services

Register your applications and folders. ColdFusion automatically registers CFCs found in the registered folders.

Add/Edit REST Service

Root path
Application path or root folder where CFCs reside

Host
Host name for the REST service. Example: localhost:8500 (Optional)

Service Mapping
Alternate string to be used for application name while calling REST service. Example: http://localhost/rest/{service mapping}/test (Optional)

Set as default application
Set an application as default to exclude the application name in the URL while calling the web service. One default application is allowed for a host. Example http://localhost/rest/path

1 Enter the root path where REST CFCs are available in your system. Alternatively you can click **Browse Server** and choose the path where the CFC application resides.

2 Enter the host name for the REST service. For example, localhost:8500

3 Enter the **Service Mapping** string name. For example, http://localhost/rest/{service mapping}/test

4 Select the check box if you want to set the application as default while calling the web service.

Access swagger api-docs

The Swagger representation of the API comprises two file types:

The Resource Listing - This root document contains general API information and lists the resources. Each resource has its own URL that defines the API operations on it.

The API Declaration - This document describes a resource, including its API calls and models.

You can verify the ColdFusion generated swagger APIs document by using the ColdFusion server path as follows:

<ColdFusion server URL path:port number>/<Service Mapping name>/api-docs/<resourcePath name>

You can access resource listing by using the same path as above without the resourcePath name. (<ColdFusion server URL path:port number>/Service Mapping name>/api-docs)

Service Mapping name is the name that you specify while registering your REST application in ColdFusion server.

For example, localhost:8500/test/api-docs/studentService

The swagger API document generated from the sample studentservice.cfc REST CFC file appears as shown in the following api document:

```
\{
   "swaggerVersion":"1.2",
   "apiVersion":"1.0",
   "basePath":"localhost:8500/rest/test",
   "resourcePath":"/studentService",
   "apis":[
       {
           "path":"/studentService/",
           "description":"",
           "operations":[
               {
                   "nickname":"addStudents",
                   "method":"POST",
                   "summary":"add students",
                   "type":"void",
                   "parameters":[
                       {
                           "name":"name",
                           "paramType":"body",
                           "allowMultiple":false,
                           "required":true,
                           "type":"array",
                           "items":{
                               "$ref":"student"
                           }
                       }
                  ]
               },
               {
                   "nickname":"deleteStudent1",
                   "method":"DELETE",
                   "summary":"delete students",
                   "type":"void",
                   "parameters":[
                       {
                           "name":"students",
                           "paramType":"body",
                           "allowMultiple":false,
                           "required":true,
                           "type":"array",
                           "items":{
                               "$ref":"student"
                           }
                       }
                  ]
               },
               {
                   "nickname":"addStudent",
                   "method":"PUT",
                   "summary":"add student",
                   "type":"void",
                   "parameters":[
                       {
                           "name":"name",
                           "paramType":"form",
                           "allowMultiple":false,
                           "required":true,
                    "type":"string"
                },
                {
                    "name":"age",
                    "paramType":"form",
                    "allowMultiple":false,
                    "required":true,
                    "type":"number"
                }
            ]
        }
    ]
},
{
    "path":"/studentService/{name}-{age}",
    "description":"",
    "operations":[
        {
            "nickname":"getStudent",
            "method":"GET",
            "summary":"retrieve student",
            "type":"student",
            "produces":[
                "application/json"
            ],
            "parameters":[
                {
                    "name":"name",
                    "paramType":"path",
                    "allowMultiple":false,
                    "required":true,
                    "type":"string"
                },
                {
                    "name":"age",
                    "paramType":"path",
                    "allowMultiple":false,
                    "required":true,
                    "type":"string"
                }
            ],
            "responseMessages":[
                {
                    "code":404,
                    "message":"Not found"
                },
                {
                    "code":200,
                    "message":"Successfull",
                    "responseModel":"student"
                }
            ]
        }
    ]
},
{
    "path":"/studentService/{studentId}",
            "description":"",
            "operations":[
                {
                    "nickname":"updateStudentAddress",
                    "method":"POST",
                    "summary":"modify student address",
                    "notes":"modify the address for given studentId",
                    "type":"address",
                    "parameters":[
                        {
                            "name":"studentId",
                            "paramType":"path",
                            "allowMultiple":false,
                            "required":true,
                            "type":"number"
                        },
                        {
                            "name":"address",
                            "paramType":"body",
                            "allowMultiple":false,
                            "required":true,
                            "type":"address"
                        }
                    ]
                }
            ]
        }
    ],
    "models":{
        "address":{
            "id":"address",
            "description":"this is a address component",
            "required":[
            ],
            "properties":{
                "country":{
                    "type":"string"
                },
                "street":{
                    "type":"string"
                },
                "houseNo":{
                    "type":"number",
                    "format":"double"
                },
                "state":{
                    "type":"string"
                }
            }
        },
        "student":{
            "id":"student",
            "description":"this is a student component",
            "required":[
                "address",
                "name",
                "age"
            ],
            "properties":{
                "address":{
                    "$ref":"IndiaAddress"
                },
                "name":{
                    "type":"string"
                },
                "age":{
                    "type":"number",
                    "format":"double"
                }
            }
        },
        "IndiaAddress":{
            "id":"IndiaAddress",
            "description":"India address fromat",
            "required":[
            ],
            "properties":{
                "country":{
                    "type":"string"
                },
                "pin":{
                    "type":"number",
                    "format":"double"
                },
                "street":{
                    "type":"string"
                },
                "district":{
                    "type":"string"
                },
                "houseNo":{
                    "type":"number",
                    "format":"double"
                },
                "state":{
                    "type":"string"
                }
            }
        }
    }
}
```

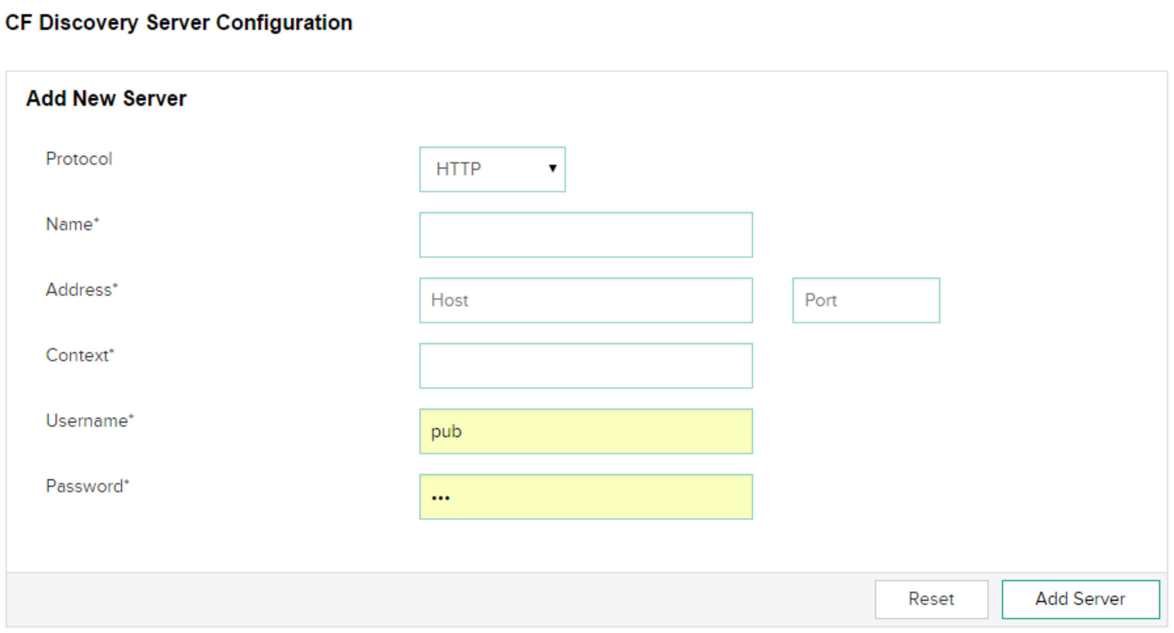
Import API from CF REST services

As a Publisher, during API creation you can import CF REST API services. Before choosing this option, ensure that you add the CF server in API Manager.

Note: *Add the CF server only when you are accessing API Manager remotely. For in-VM access, API Manager detects the REST services automatically and loads them.API Manager can be installed over ColdFusion in your system. API Manager is installed in cfusion/ApiManager path. For more information, refer to ColdFusion and API Manager - Integrationfeature.*

Add CF server in API Manager

Start API Manager in Administrator mode. Click **CF Discovery Server Configuration** on the left pane. Add the configuration values as shown in the sample snapshot below.



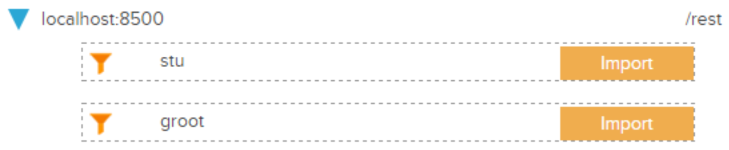
CF Discovery Server Configuration

Add New Server

Protocol	HTTP	
Name*		
Address*	Host	Port
Context*		
Username*	pub	
Password*	•••	

Import API

Start API Manager portal. Click **Create New API** and click **Import REST API from ColdFusion**.



localhost:8500 /rest

stu

groot

CFC and Swagger mapping structure

You can compare CFC field types and Swagger field types from the following mapping structures.

Resource listing schema

The Resource Listing serves as the root document for the API description. It contains general information about the API and an inventory of the available resources.

Swagger doc Field Name	Type	Description	CF Fields
SwaggerVersion	String	**Required.** Specifies the Swagger Specification version being used.	Update programmatically using API Manager
apis	Resource Object	**Required.** Lists the resources of the specification. The array can have 0 or more elements	N/A

apiVersion	string	Provides the version of the application API	Modify using application.cfc file
info	Info Object	Provides metadata about the API. The clients can use this metadata, and can be presented in the Swagger-UI for convenience.	Modify using application.cfc file
authorizations	Authorizations Object	Provides information about the authorization schemes allowed on this API. The type of the authorization scheme. Values MUST be either"basicAuth", "apiKey", or "oauth2".	Update programmatically using API Manager

API declaration schema

The API declaration provides information about an API exposed on a resource. Specify only one file per resource. Save the file in the URL described by the path field.

Swagger doc **Field Name**	**Type**	**Description**	**CF Field**
basePath	string	**Required**. The root URL serving the API.	Add programmatically while parsing CFC
consumes	[string]	A list of MIME types the APIs on this resource can consume. The MIME types are global to all APIs but can be overridden on specific API calls.	Cfcomponent.consumes
produces	[string]	A list of MIME types the APIs on this resource can produce. The MIME types are global to all APIs but can be overridden on specific API calls.	Cfcomponent.produces
resourcePath	string	The relative path to the resource, from the basePath, which this API Specification describes.	Cfcomponent.restpath
apis	[API Object]	**Required**. A list of the APIs exposed on this resource. There MUST NOT be more than one API Object per path in the array.	Details in API Object
apiVersion	string	Provides the version of the application API (not to be confused by the (specification version).	N/A

swaggerVersion	string	**Required.** Specifies the Swagger Specification version being used.	N/A
authorizations	Authorizations Object	A list of authorizations schemes required for the operations listed in this API declaration. Individual operations override this setting. If there are multiple authorization schemes described here, it means they're all applied.	Add programmatically as API Manager updates Authorization info
models	Models Object	A list of the models available to this resource. Expose the models separately for each API Declaration.	Generate programmatically

API object schema

The API Object describes one or more operations on a single path. In the apis array, there MUST be only one API Object per path.

Swagger doc Field Name	Type	Description	CF Field
description	String	A short description of the resource.	Cffunction.description
operations	[Operation Object]	**Required.** A list of the API operations available on this path. The array includes zero or more operations.	Details in Operation object schema
Path	String	**Required.** The relative path to the operation, from the basePath, which this operation describes. The value SHOULD be in a relative (URL) path format.	Component.restpath + Cffunction.restpath

Operation object schema

The Operation Object describes a single operation on a path. In the operations array, there must be only one Operation Object per method. This object includes the Data Type Fields to describe the return value of the operation. The type field must be used to link to other models.

Swagger doc Field Name	Type	Description	CF Field
authorization	Authorizations Object	A list of authorizations required to execute this operation	Programmatically from API Manager
consumes	[string]	A list of MIME types this operation can consume.	Cffunction.consumes
method	String	**Required.** The HTTP method required to call the operation.The value MUST be one of the following values: "GET", "HEAD", "POST", "PUT","PATCH", "DELETE", "OPTIONS". The values MUST be in uppercase.	Cffunction. httpmethod

nickname	String	**Required.** A unique id for the operation that is used by tools reading the output for further and easier manipulation	Cffunction.name
notes	String	A verbose explanation of the operation behavior.	Cffunction.hint
parameters	[Parameter Object]	**Required.** The inputs to the operation. If no parameters are needed, an empty array MUST be included.	Details in Parameter object schema
produces	[string]	A list of MIME types this operation can produce.	Cffunction.produces
responseMessages	[Response Message Object]	Lists the possible response statuses that can return from the operation.	New parameter introduced in Cfunction
summary	String	A short summary of what the operation does. For maximum readability in the swagger-ui, this field SHOULD be fewer than 120 characters.	Cffunction.description

Parameter object schema

The Parameter Object describes a single parameter to be sent in an operation and maps to the parameters field in the Operation Object. This object includes the Data Type Fields to describe the type of this parameter. The type field must be used to link to other models.

If type is File, the consumes field must "multipart/form-data", and the paramType must be "form".

Swagger doc Field Name	**Type**	**Description**	**CF Field**
allowMultiple	boolean	Another way to allow multiple values for a "query", "header" or "path" parameter.	Not available in ColdFusion.
description	string	**Recommended.** A brief description of this parameter.	Cfargument.hint
name	string	**Required.** The unique name for the parameter.	Cfargument.name
paramType	string	**Required.** The type of the parameter.The value MUST be one of these values: "path", "query", "body","header", "form" Note: As per spec swagger dosen't support "Cookie", "Matrix" paramtype in ColdFusion	Cfargument. restargsource
required	boolean	A flag to note whether this parameter is mandatory.	Cfargument.required

CFC/Swagger/Java types comparison

CFC	Swagger	Java	Additional information
string	string	string	
uuid	string	string	
guid	string	string	
query	custom model	coldfusion.xml.rpc.DocumentQueryBean	
void	void		for argument map to "body"
numeric	number(format double)	Double	
boolean	boolean	boolean	
date	string(format date)	java.util.Calendar	
any	object	java.lang.Object	
array	array of objects	java.lang.Object[]	
binary	array of byte	byte[]	
struct	custom model	java.util.Map	
xml	string	org.w3c.dom.Documents	

Authentication types

Overview

In the API Manager, you can use API keys to authenticate your APIs and applications. The API Manager generates the API keys and enable you to add API key-based authentication to your APIs.

Validation using API keys is a type of security you can enforce while creating an API. Applications use the API key and the API Manager checks to see if the API key is in an approved state for a resource.

The API Manager uses three API authentication types:

1 apiKey

2 basicAuth

3 OAuth2

OAuth2

In any traditional client-server application when the client requests for a protected resource or web page, the server authenticates the client. The client passes the credentials to the server and the authentication happens. Based on the authentication result, the client gets access to the protected resource. This approach works fine in a controlled environment where an organization/entity manages both the client and server applications.

Today's organizations adopt an open and collaborative platform architecture in which the data is shared through an API mechanism. The API mechanism allows any developer (internal or customers) to quickly interact with the service without the intervention of the service owner. This framework enables users to build mashups, mobile and desktop client applications, and other third-party services.

The services often require access to sensitive data of an API which in turn requires authentication or access control. One common method of authentication is the user name and password authentication. However, providing password to a third-party application has many disadvantages. Entering a password in a third-party application requires a user's trust. Also, from a usability perspective, it is not possible to revoke an application until the password has changed. In the same way, the application fails to work reliably if the password has changed.

OAuth2 combats the risks mentioned above, that are associated with the password anti-pattern. OAuth2 is a standard for delegating authorization to an application on behalf of the user without providing the password. Instead of the user password, an application receives an access token with limited lifetime access. Depending on the type of application, obtaining an access token varies. These access tokens are called grant types. To know more about OAuth 2.0, refer to IETF OAuth 2.0 specification.

OAuth2: Terms and actors

Resource owner

An entity capable of granting access to a protected API. A resource owner can be an end user. The resource owner delegates authorization to the third-party client or application. Through this delegation, the application can access the API on behalf of resource owner.

Resource Server

The server hosting the APIs. The resource server accepts and responds to protected API requests using access tokens. The API Manager acts as resource server that validates access tokens and performs access-control on a protected API.

Client

An application that makes protected resource requests on behalf of the resource owner and the owner's authorization. The term "client" does not imply to any particular implementation characteristics (for example, if the application executes on a server, a desktop, or other devices). The API Manager application is an OAuth2 client.

Authorization Server

The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization. The API Manager acts as an authorization server as well.

Access Token

Access tokens are credentials to access protected APIs. An access token is a string representing an authorization issued to the client. The string is opaque to the client. Tokens represent specific scopes and durations of access, granted by the resource owner, and enforced by the resource server and authorization server. The API Manager currently supports Bearer Access tokens which are opaque tokens.

Refresh Token

Refresh tokens are credentials used to obtain access tokens. The authorization server issues the refresh tokens to the client. The client obtains refresh tokens when the current token becomes invalid or expires.

Authorization Code

An opaque code obtained through authorization code grant flow can be redeemed to get a limited life time access token and a refresh token. Once code is redeemed, it is no longer valid and cannot be used to get access tokens.

Scope

A scope is a permission associated with a protected resource. If the publisher protects any resource using OAuth2, specifying a scope along with it requires the access token to have this scope associated with it.

Client ID

Every application with OAuth2 authorization has a unique identifier.

Client Secret

The Client Secret gets the access token in Client credentials flow and authorization code flow.

The API Manager provides complete OAuth2 support including authentication provider, role-based authorization framework for scopes, and login web pages along with token management component. The API Manager acts as authorization server and resource server. The authentication provider supports integration with LDAP, database, and federated authentication using SAML.

OAuth2 endpoints

The API Manager OAuth2 provider provides the following endpoints which are used to obtain and manage access tokens:

Authorization Endpoint

http(s)://servername:serverport/oauth2/auth: The endpoint where the end user logs in and provides the consent to provision the authorization grant to the application.

Token Endpoint

http(s)://servername:serverport/oauth2/token: The endpoint that the client uses to obtain an access token by presenting its authorization grant or refresh token.

Redirect Endpoint

After the user interacts with the authorization endpoint, a user is redirected back to the application. The redirect (or callback) URL is specified in the application created in the API Manager. Currently, the API Manager does not support multiple redirect URL's registration in the application.

Token Info Endpoint

http(s)://servername:serverport/oauth2/tokeninfo: The tokeninfo endpoint provides information about the access token. The response of the endpoint includes information like create time, expiry, Application Id (client_id) of the logged in user, and scopes.

The token info endpoint receives the access token in two ways:

1. Using query parameter access_token

2. Using the authorization header

Example of a simple HTTP GET request with query parameter access _token:

http(s)://example.com/oauth2/tokeninfo?access_token=3ffb313f16856a4d6b1feecd2e50b950

(or) using the authorization header:

GET example.com/oauth2/token HTTP/1.1

Host: example.com

Authorization: Bearer 3ffb313f16856a4d6b1feecd2e50b950

Token Revocation Endpoint

http(s)://servername:serverport/oauth2/revoke: The endpoint that can be used to revoke any access token or refresh token. Once revoked, the access token or refresh token becomes invalid.

OAuth2 Self Service

http(s)://servername:serverport/oauth2/self/<Logon Identifier>: If the OAuth2 server authentication provider is a user store (LDAP or database), an end user can view the list of approved applications along with the approved scopes. The end user can also revoke any application access. Configure the login identifier in the user store to set up this portal.

Application Registration

To obtain access tokens or authorization codes, register the application with the API Manager. The API Manager authorization server identifies the application. Registering an application with the API manager provides the following pair of keys:

- client Id: Client Id is a public identifier
- client secret: Client secret is a secret identifier

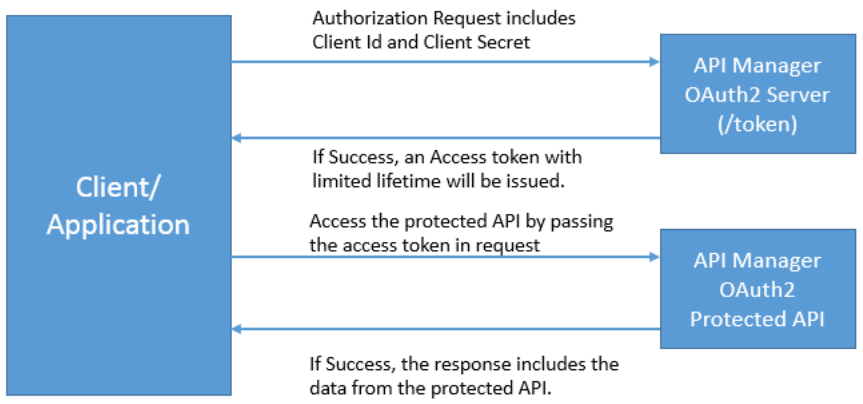
Register the callback URL (or redirect URL) of the application with the API Manager so that the deployed application can access the token. To use any of OAuth2 grant types, the API Manager subscribes to at least one API that is protected using OAuth2.

Grant types

A grant is a method of acquiring an access token that accesses an OAuth2- protected API. OAuth2 provides different grant types for different scenarios based on the type of application (server, browser, or mobile).

Client credentials grant type

Among the four grant types of OAuth 2.0, client credentials is the simplest grant type. This grant type can be used in cases where an application that is confidential (means ability to store client credentials safely) consumes the API.


Client/ Application
Authorization Request includes Client Id and Client Secret
API Manager OAuth2 Server (/token)
If Success, an Access token with limited lifetime will be issued.
Access the protected API by passing the access token in request
API Manager OAuth2 Protected API
If Success, the response includes the data from the protected API.

Prerequisites

Register an application with the API Manager. The application subscribes to at least one API that is protected using OAuth2.

Access Token Request

In this flow/grant type, the following request is made to the token endpoint with the following details. The HTTPs request is made using HTTP POST with the content-type “application/x-www-form-urlencoded”.

http(s)://servername:serverport/oauth2/token

Request Parameter	Description of the parameter
grant_type	Specifies the requested grant type. To use this flow, the value is **client_credentials.**
client_id	The Id that uniquely identifies the application. The client id can be found in the API Manager application.
client_secret	Specifies the client credential registered with the application in API Manager.
scope (optional)	Specifies the list of scopes required for the access token. If multiple scopes are required, specify the scopes as a space-delimited string.

The sample below is an HTTP request to request an access token with scopes:

POST example.com/oauth2/token HTTP/1.1

Host: example.com

Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_id=625bc9f6-3bf6-4b6d-94ba-e97cf07a22de&client_secret=625bc123-3bf6-4b6d-94ba-e97cf07a22de&scope=sample_read%20sample_write

Both client Id and client secret can also be sent using the authorization header. The sample HTTP request requests an access token by passing the client credentials in authorization header along with scopes.

Authorization: Basic Base64 (clientId:clientSecret)

POST example.com/oauth2/token HTTP/1.1

Host: server.example.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&scope=sample_read%20sample_write

Access Token Response

If the request for the access token is successful, the response is sent in JSON Format. The JSON includes the following keys.

Response Parameter	Description of the Response Parameter
access_token	The access token that can be used to call the protected API.
token_type	Specifies the type of the requested access token. Currently the API Manager supports only bearer tokens.
expires_in	The lifetime in seconds of the access token. The access token lifetime can be configured by changing the access token lifetime in API Manager application.
scope (optional)	The Granted/Approved scopes associated with the requested access token.

The following example shows a JSON response:

```
{
 "access_token": "7ee85874dde4c7235b6c3afc82e3fb",
 "token_type": "bearer",
 "expires_in": 1200,
 "scope": " sample_read sample_write"
}
```

To retrieve this information any time, the client application can pass the requested access token to the token info endpoint. As this flow involves direct client authentication, the endpoint response does not include any user info.

Access Token Revocation

Every access token has an expiry time. The access token becomes invalid and revoked automatically when the token reaches its expiry time. When the token expires, applications request for a new token and is granted one. Client credentials flow/grant type does not grant any refresh token. If the access token has to be revoked before its expiry time, pass the access token to the revocation endpoint. When revoking the access token using client credentials, the flow requires passing the client credentials in the token revocation request for a second time.

The access token revocation request is sent to the API Manager OAuth2 revocation endpoint. The following sample HTTP request shows how to revoke the Access token issued using client credentials flow.

```
POST example.com/oauth2/revoke HTTP/1.1

Host: example.com

Content-Type: application/x-www-form-urlencoded

token=7ee85874dde4c7235b6c3afc82e3fb&token_type_hint=access_token&client_id=625bc9f6-3bf6-4b6d-94ba-
e97cf07a22de&client_secret=625bc123-3bf6-4b6d-94ba-e97cf07a22de
```

Client Secret Regeneration

You can generate the Client Secret by logging into the portal to prevent any unauthorized access to the Client Secret.

Once the Client Secret is regenerated, the older Client Secret becomes invalid. Use the new Client Secret to request for new access tokens.

Security Considerations

The security of OAuth2 depends on the secure transmission of OAuth2 access tokens and client credentials. Therefore, access all the OAuth2 endpoints on **HTTPS** (SSL/TLS) transport for production environments.

You can choose **Mandate HTTPS** in the Security Configuration page in the API Manager administrator to prevent all OAuth2 endpoints from accepting any request over HTTP. Keep the client secret confidential and store it in a secure database.

Keep the application private, otherwise the credentials can be captured by decompiling/analyzing the source code of the application. Use this flow on server-side application where secure storage can be guaranteed.

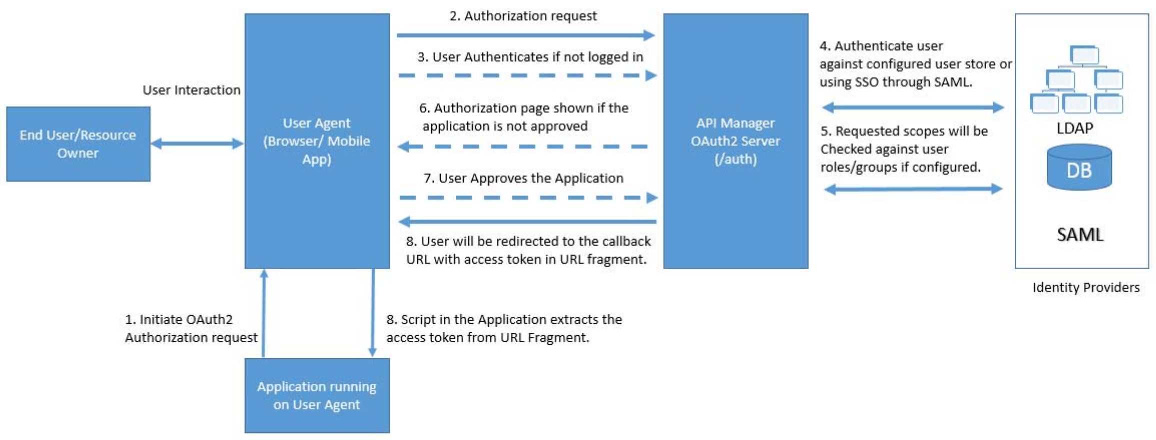
How is Client Credential grant different from API Keys security?

In case of client credentials, you use the access token is used to call the API. On the other hand, in API Key security, you can use the API key to validate the API. You get an extra layer of security control where you can revoke the access token easily without modifying the client application.

Implicit/token grant type (two-legged OAuth2)

Implicit/token grant is one of the browser-based redirection flows that is suitable for pure JavaScript (client-side)-centric or mobile applications that cannot store user credentials and does not have any server side component.

A simple example is a pure JavaScript application made using Angular/Ember and calls the API directly from JavaScript via Ajax calls. This grant type/flow requires interaction with the end user.


End User/Resource Owner
User Interaction
User Agent (Browser/ Mobile App)
2. Authorization request
3. User Authenticates if not logged in
6. Authorization page shown if the application is not approved
7. User Approves the Application
8. User will be redirected to the callback URL with access token in URL fragment.
API Manager OAuth2 Server (/auth)
4. Authenticate user against configured user store or using SSO through SAML.
5. Requested scopes will be Checked against user roles/groups if configured.
LDAP
DB
SAML
Identity Providers
1. Initiate OAuth2 Authorization request
8. Script in the Application extracts the access token from URL Fragment.
Application running on User Agent

Prerequisites

Register an application with the API Manager. The application subscribes to at least one API that is protected using OAuth2. Specify the callback URL (or redirection URL) in the API Manager. After authorization request is completed, you are redirected to this registered URL. The registered application includes the client ID that is required for this flow/grant type.

Requesting an Access Token

The flow begins by the application by redirecting the user to API Manager OAuth2 server authorization endpoint (/oauth2/auth) when the application tries to access an OAuth2 protected API. The URL for sending the authorization request is shown as below:

http://<APIM_SERVERNAME>:port/oauth2/auth?client_id=<Application Client ID>&response_type=token&scope=<list of scopes delimited by string>&state=<random string>

For example,

http://example.com/oauth2/auth?client_id=9a42a56d5b5546079f2f82a62612dab9&response_type=token&

state=nkj34898sdcsd123&scope=foo_read%20foo_write

Request Parameter	Description of the parameter
response_type	This parameter identifies this grant type flow. The value is **token**.
client_id	This parameter uniquely identifies the application. The client id can be found in the API Manager application.
state	This parameter is an opaque pseudo-random string value used by the client application to maintain state between the initial authorization request and callback. The parameter prevents cross-site request forgery. Store the scope locally when the flow starts after which can be used later for comparison.
Scope (optional)	Specifies the list of scopes required for the access token. If multiple scopes are required, specify the scopes as a space delimited string.

Once the API Manager OAuth2 server receives the request, it validates all the parameters. If valid, and the authentication method for application is user-store, you see a login page. If the authentication method for the application is SAML SSO, you are redirected to the SAML Identity provider for authentication. After the authentication is successful, you are presented with another page where you can provide the authorization/consent.

Here, you can select the list of scopes that are permitted for the application, which in turn is provided to the access token.

Once it is redirected to the callback URL, the application has a script to extract the access token out of the URL fragment. The server does not receive the access token because the URL fragment is not sent over HTTP/S by browsers. The callback URL's fragment contains the following parameters:

URL Fragment Parameter	Description of the parameter
access_token	The requested access token that can be used to call the protected API.
token_type	Specifies the type of the requested access token. Currently the API Manager supports bearer tokens only so the value is Bearer.
expires_in	The lifetime, in seconds, of the access token. The access token lifetime can be configured by changing the access token lifetime in API Manager application.
client_id	Client id of the application for which the access token is issued.
state	An opaque pseudo-random string value used by the client application to maintain state between the initial authorization request and callback. The parameter is used to prevent cross-site request forgery.
scope (optional)	The granted/approved scopes associated with the requested access token.

Once the access token is obtained from the grant type, it calls the protected API and gets the response from the API.

To retrieve this information any time, the client application can pass the requested access token to the token info endpoint. The API Manager OAuth2 authorization server does not issue any refresh tokens. Once the issued access token expires, the application goes through the OAuth2 two-legged process again.

The API Manager admin portal controls the timeout for the login credentials. If the authentication times out, provide valid credentials (user name and password) again to log in in case of user store authentication method. In case of SAML SSO IDP, authorization happens again. If the application is already approved, you are directly redirected to the callback URL with the fragment having the access token and other details.

Security Considerations

The security of OAuth2 depends on the secure transmission of user credentials, OAuth2 Access Tokens, and Client Credentials. Therefore accessing all the OAuth2 endpoints on **HTTPS** (SSL/TLS) transport is recommended for production environments. The API Manager administrator can mandate HTTPS transport only in the Security Configuration page.

Also, register the HTTPs URL callback with the API Manager because the callback URL receives the access token. The application verifies the state parameter value received in the fragment with the value stored in the initial request. Also, verify that the client id from the URL fragment is same as the client id in the JavaScript application code.

Access Token Revocation

Every access token has an expiry time. Once it reaches the expiry time, the access token becomes invalid and is revoked automatically. Once the access token expires, the application makes a request for the access token again. If the access token has to be revoked before its expiry time, pass the access token to the revocation endpoint.

The access token revocation request is sent to the API Manager OAuth2 revocation endpoint. The following sample HTTP request shows how to revoke the access token:

POST example.com/oauth2/revoke HTTP/1.1

Host: example.com

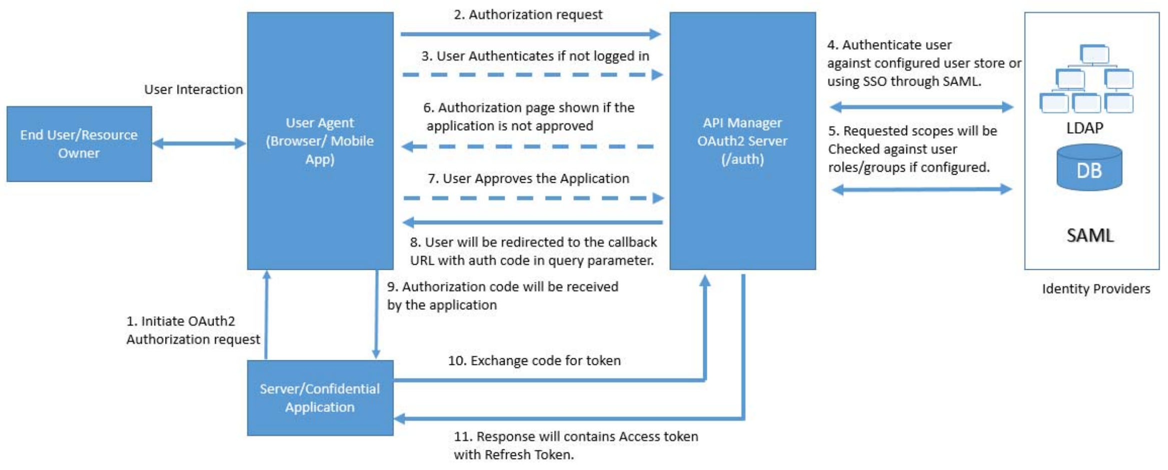
Content-Type: application/x-www-form-urlencoded

token=7ee85874dde4c7235b6c3afc82e3fb&token_type_hint=access_token

Authorization code grant type (three-legged OAuth 2)

This grant type is suitable for server applications. In this flow, the access token is hidden from the end user, and instead issues an authorization code.

Redeem the intermediary authorization code at the token endpoint with client credentials to receive access token and refresh token. This flow also requires application to have a server component where it redeems the authorization code for accessing the token and refreshing the token. Once the code is redeemed, it is no longer valid.


2. Authorization request
3. User Authenticates if not logged in
User Interaction
End User/Resource Owner
User Agent (Browser/ Mobile App)
6. Authorization page shown if the application is not approved
7. User Approves the Application
8. User will be redirected to the callback URL with auth code in query parameter.
API Manager OAuth2 Server (/auth)
4. Authenticate user against configured user store or using SSO through SAML.
5. Requested scopes will be Checked against user roles/groups if configured.
LDAP
DB
SAML
Identity Providers
9. Authorization code will be received by the application
1. Initiate OAuth2 Authorization request
10. Exchange code for token
Server/Confidential Application
11. Response will contains Access token with Refresh Token.

Prerequisites

Register an application with the API Manager and subscribe to at least one API that is protected using OAuth2. Also, specify a callback URL (or redirection URL) with the API Manager application so that after authorization request is completed user is redirected to this registered URL. The registered application includes Client ID and Client Secret, which are required for this flow/grant type.

Requesting an access token

The flow begins by the application server component redirecting the end user to API Manager OAuth2 server authorization endpoint (/oauth2/auth). The URL for sending the authorization request is shown below:

http://<APIM_SERVERNAME>:port/oauth2/auth?client_id=<Application Client ID>&response_type=code&scope=<list of scopes delimited by string>&state=<random string>

An example URL is:

http://example.com/oauth2/auth?client_id=9a42a56d5b5546079f2f82a62612dab9&response_type=code&

state=nkj34898sdcsd123&scope=foo_read%20foo_write

Request Parameter	**Description of the parameter**
response_type	The parameter identifies the flow of the grant type. The value is code.
client_id	The Client Id of the application which uniquely identifies the application. The client id can be found in the API Manager application.
state	An opaque pseudo- random string value used by the client application to maintain state between the initial authorization request and callback. The parameter is used to prevent cross-site request forgery. Store the state in a server-side session for comparing it later.
scope (optional)	Specifies the list of scopes required for the access token. If multiple scopes are required, specify the scopes as a space delimited string.

Once the API Manager OAuth2 server receives this request, it validates all the parameters. If the parameters are valid, the user sees a login page where the user is asked to provide the login credentials (user name and password). If the authentication method for the application is SAML SSO, the user is redirected to the SAML Identity provider for authentication.

After the authentication is successful, you are presented with another page where you can provide the authorization/consent. Here you can select the list of scopes that are permitted for the application, which in turn is provided to the access token. Once scopes are selected and approved, you are redirected (to the registered URL) back to the application. You attach a query parameter called code with its value along with the state parameter.

Now the application server component verifies the state parameter value received from the request with the value stored in the session. If state parameter is verified, then extract the code parameter value from the request and use client credentials to redeem this code for an access token.

POST example.com/oauth2/token HTTP/1.1

Host: server.example.com

Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&client_id=9a42a56d5b5546079f2f82a62612dab9&

client_secret=7ee85874dde4c7235b6c3afc82e3fb

If the given client credentials & the authorization code are valid the response will contain an access token along with a refresh token. Refresh token is used to get a new access token when the current access token is expired/revoked. The JSON response includes the following keys.

A sample JSON response is shown below.

```
{
"access_token": "95d9c3de53a9c48e629ecb6a288f6c",
"token_type": "bearer",
"expires_in": 1200,
"scope": "foo_read",
"refresh_token": "31cf059933238e866779a43237cd7ec"
}
```

JSON Key	**Description of the Key**
access_token	The requested access token which can be used to call the protected API.
token_type	Specifies the type of the requested access token. Currently API Manager supports bearer tokens only so the value is Bearer.
expires_in	The lifetime in seconds of the access token. The access token lifetime can be configured by changing the access token lifetime in API Manager application.
client_id	Client Id of the application for which the access token is issued.
State	An opaque pseudo- random string value used by the client application to maintain state between the initial authorization request and callback. The parameter is used to prevent cross-site request forgery.
scope	The Granted/Approved scopes associated with the requested access token.
refresh_token	Refresh token is a long-lived access token which is used to ask for a new access token.

To retrieve this information any time, the client application can pass the requested access token to the token info endpoint. Once the issued access token expires, the application can use the refresh token to get a new access token.

Refreshing access token

The example below shows how to refresh an access token using refresh token. Pass the client credentials along with the request, either through form parameters or using basic authentication.

```
POST example.com/oauth2/token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded
grant_type=refresh_token&refresh_token=31cf059933238e866779a43237cd7ec&scope=foo_read
```

If the refresh token is expired/revoked, the application starts the three-legged dance once again. In case of user store authentication, if the authentication times out, provide valid credentials (user name and password) again to log in. In case of SAML SSO IDP, authorization happens again. If the application is already approved, you are redirected to the callback URL immediately with the intermediary authorization code and state.

Access/refresh token revocation

Each issued access token has an associated expiry time. Once the expiry time is reached, the access token becomes invalid & revoked automatically. Once the access token expires, the application gets a new access token. If the access token has to be revoked before its expiry time, pass the access token to the revocation endpoint.

The access token revocation request is sent to the API Manager OAuth2 revocation endpoint. The following sample HTTP request revokes the access token:

POST example.com/oauth2/revoke HTTP/1.1

Host: example.com

Content-Type: application/x-www-form-urlencoded

token=7ee85874dde4c7235b6c3afc82e3fb&token_type_hint=access_token

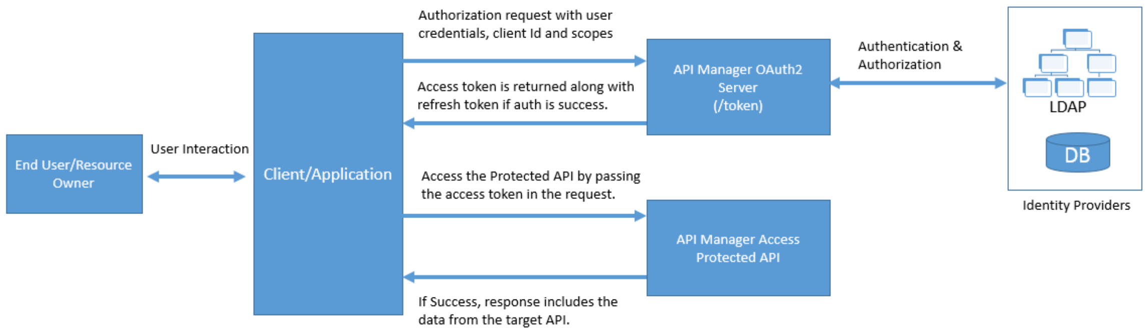
The revoke endpoint also supports revoking the refresh token pass the refresh token in token parameter and set token_type_hint parameter value as refresh_token.

Security considerations

The security of OAuth2 depends on the secure transmission of user credentials, OAuth2 access tokens, and client credentials. Therefore, accessing all the OAuth2 endpoints on **HTTPS** (SSL/TLS) transport is recommended for production environments. The API Manager administrator can mandate HTTPS transport only in the Security Configuration page. Also, HTTPS URL callback is registered with the API Manager application because the callback URL receives the authorization code. The server application verifies the state parameter value received in the request with the value stored in the session. The server application securely stores the access tokens, refresh tokens, and client credentials in a secure database.

Resource owner/password grant type

This grant type allows the application to have their own login page. The password grant type is used in the cases where you have complete trust on the application. This grant type is suitable for a trusted mobile application or a desktop application. This grant type requires user interaction for requesting the access token.


Authorization request with user credentials, client Id and scopes
Access token is returned along with refresh token if auth is success.
API Manager OAuth2 Server (/token)
Authentication & Authorization
LDAP
DB
Identity Providers
End User/Resource Owner
User Interaction
Client/Application
Access the Protected API by passing the access token in the request.
API Manager Access Protected API
If Success, response includes the data from the target API.

Prerequisites

Register an application API Manager. It is mandatory that the application subscribes least one API that is protected using OAuth2. The registered application includes Client ID which is required for this flow/grant type. For this grant type to function, it is important that all subscribed APIs of type User store are configured with the same user store.

Requesting an Access Token

The user initiates the flow when accessing the installed application is presented with a login form provided by the application. User enters the login credentials (user name, password) then the application sends an authorization request to the API manager OAuth2 server.

In this flow/grant type, the following request is made to the token endpoint with the following details. The HTTPs request is made using HTTP POST having the content-type "application/x-www-form-urlencoded".

http(s)://servername:serverport/oauth2/token

Request Parameter	**Description of the parameter**
grant_type	Specifies the requested grant type. To use this flow, the value is password**.**
client_id	Client Id of the application which uniquely identifies the application. The client id can be found in the API Manager application.
user name	End user provided user name
password	End user provided password
scope (optional)	Specifies the list of scopes required for the access token. If multiple scopes are required, specify the scopes as a space delimited string.

POST /oauth2/token HTTP/1.1

Host: example.com

Content-Type: application/x-www-form-urlencoded

grant_type=password&username=maxwell&password=sdcoio2380&client_id=
95d9c3de53a9c48e629ecb6a288f6c&scope=foo_read%20foo_write

The API Manager Server validates the incoming request and checks whether the client id is valid and is allowed to use the password grant type. Then it checks for the application authentication method is user store or not and authenticates the given user name and password against the user store. If the authentication is successful, the Oauth2 server validates the scopes requested.

If the requested scopes are not empty, each scope is looked at whether granting this scope requires any group/role membership. If user is part of the roles specified in the scope, that scope is approved. An access token with approved scopes along with a refresh token is returned as part of the response. The JSON response includes the following keys.

JSON Key	**Description of the Key**
access_token	The requested access token which can be used to call the protected API.
token_type	Specifies the type of the requested access token. Currently API Manager supports bearer tokens only so the value is Bearer.

expires_in	The lifetime of the access token in seconds. The access token lifetime can be configured by changing the access token lifetime in API Manager.
scope	The granted/approved scopes associated with the requested access token.
refresh_token	Refresh token is a long-lived access token which is used to ask for a new access token.

A sample JSON response is shown below:

```
{
"access_token": "95d9c3de53a9c48e629ecb6a288f6c",
"token_type": "bearer",
"expires_in": 2800,
"scope":"foo_read foo_write",
"refresh_token": "31cf059933238e866779a43237cd7ec"
}
```

To retrieve this information any time, the client application can pass the requested access token to the token info endpoint. Once the issued access token expires, the application can use the refresh token to get a new access token.

Refreshing Access Token

The example below shows how to refresh an access token using refresh token.

```
POST example.com/oauth2/token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type=refresh_token&refresh_token=31cf059933238e866779a43237cd7ec&scope=foo_read
```

If the refresh token is expired/revoked, the application asks you for the credentials. Restart the complete flow again.

Access/refresh token revocation

Each issued access token has an associated expiry time. Once the expiry time is reached, the access token becomes invalid and revoked automatically. Once access token expires, the application gets a new access token. If the access token has to be revoked before its expiry time, pass the access token to the revocation endpoint.

The access token revocation request is sent to the API Manager OAuth2 revocation endpoint. The following sample HTTP request shows how to revoke the Access token

```
POST example.com/oauth2/revoke HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
token=7ee85874dde4c7235b6c3afc82e3fb&token_type_hint=access_token
```

The revoke endpoint also supports revoking the refresh token pass the refresh token in token parameter and set token_type_hint parameter value as refresh_token.

Security Considerations

The security of OAuth2 depends on the secure transmission of user credentials, OAuth2 Access Tokens, and Client Credentials. Therefore accessing all the OAuth2 endpoints on **HTTPS** (SSL/TLS) transport is recommended for production environments. The client application does not store user credentials anywhere and securely stores them. If it is a desktop application, use a secure database. If it is an Android mobile application, store the access tokens and refresh tokens using SharedPreferences API. If it is iOS, store the tokens in KeyChain.

Calling an OAuth2-protected API

Once an access token is obtained from any of the grant types, it can be used to call the OAuth2-protected API. The authorization header format is:

Authorization: Bearer <OAuth2 Access Token>

Pass the access token in the Authorization Header as shown below.

GET example.com/sampleapi/v1.0/examples HTTP/1.1

Host: example.com

Authorization: Bearer 7ee85874dde4c7235b6c3afc82e3fb

Currently, the API Manager only issues bearer tokens. The API Manager validates the access token. If the token is valid, then the API Manager checks if the access token has the scope to call the protected API.

If both the conditions are true, then the API Manager calls the backend and gets the response and provides it to the client.

API Key authentication

An API Key is an opaque token, which is a simple form of authentication to consume an API. To acquire an API Key, create an application as a subscriber.

Consuming an API protected using API Key

In a request, you can pass an API Key to the API Manager in a request in one of the following ways:

- **Header:** The api_key header can be used to pass the API Key to the API runtime.
- **Query Parameter:** The api_key query parameter can be used to pass the API Key.
- **Form Parameter:** The api_key form parameter can be used to pass the API Key.

When consuming an API protected using API Key, API runtime checks for the API Key in the header. If the key is not found, then API runtime checks in the query parameters. If the key is not found, then the API runtime checks the form (or post) parameter.

If the extracted API Key is valid, the API Manager checks whether this API Key is approved or not. If the key is approved, the API manager gets the SLA plan for this subscription and then uses it for the rate limiting and throttling.

API Key regeneration

When an API Key is regenerated, the old Key of the application is no longer valid. The application starts using the new key to consume the API Key.

Basic authentication

Basic authentication is the simplest form of authenticating a user by passing the user name and password in the request when consuming an API. The API Manager allows protecting an API using the Basic authentication scheme.

A publisher can configure the user store to be used for the authentication. If the API resource is protected with a scope configured with roles, then the API Manager also authorizes the role. The authorization happens during the consumption of the API call.

Create an application in the API Manager and subscribe to an API protected using basic authentication.

With HTTP basic authentication, the user's user name and password are concatenated, base64-encoded, and passed in the Authorization HTTP header as follows:

Authorization: BASIC Base64 (username:password)

example: Authorization: Basic dm9yZGVsOnZvcmRlbA==

But simply passing the user name or password does not identify the application consuming the API. Send the application unique identifier (client ID) along with the request.

An example of consuming an API protected using BASIC authentication is as follows:

GET example.com/sampleapi/v1.0/examples HTTP/1.1

Host: example.com

Authorization: Basic dm9yZGVsOnZvcmRlbA==

clientid: 3ffb313f16856a4d6b1feecd2e50b950

Once the API Manager receives a request for an API that is protected using basic authentication, it looks for the application id. If the application id (client id) in the request is valid and approved, then the API Manager authenticates the credentials in the request.

If authentication is successful, then the API Manager checks if the current API requested is protected with appropriate scope and authorized role. If yes, the API Manager retrieves the user roles and checks if the current user has all the required roles to consume this API. Once role authorization is successful, the API Manager gets the SLA plan for this subscription for rate limiting and throttling. If rate limit is also allowed, the request is forwarded to get the response from the application.