



Address Manager

8.3.0 API Guide

November 2017 Update



CONFIDENTIAL - For customer's internal use only.

This document may not be reproduced or distributed without the written consent of BlueCat.

Legal Notices

READ THIS BEFORE INSTALLING OR USING THE PRODUCT

This Documentation is subject to the applicable BlueCat License Agreement previously entered into between BlueCat and your company, or if none, then to BlueCat's standard terms and conditions which you can view and download from <https://www.bluecatnetworks.com/services-support/support/license-agreements/>. BlueCat reserves the right to revise this Documentation at any time without notice.

Company names and/or data used in screens and sample output are fictitious, unless otherwise stated.

Copyright

©2001—2017 BlueCat Networks (USA) Inc. and its affiliates (collectively '**BlueCat**'). All rights reserved. This document contains confidential and proprietary information and is intended only for the person(s) to whom it is transmitted. Any reproduction of this document, in whole or in part, without the prior written consent of BlueCat is prohibited.

Trademarks

Proteus, Adonis, BlueCat DNS/DHCP Server, BlueCat Address Manager, BlueCat DNS Edge, BlueCat Device Registration Portal, BlueCat DNS Integrity, BlueCat DNS Integrity Gateway, BlueCat Mobile Security, BlueCat Address Manager for Windows Server, and BlueCat Threat Protection are trademarks of BlueCat.

iDRAC is a registered trademark of Dell Inc. Windows is a registered trademark of Microsoft Corporation. UNIX is a registered trademark of The Open Group. Linux is a registered trademark of Linus Torvalds. QRadar is a registered trademark of IBM. ArcSight is a registered trademark of Hewlett Packard. Ubuntu is a registered trademark of Canonical Ltd. CentOS is a trademark of the CentOS Project. All other product and company names are registered trademarks or trademarks of their respective holders.

Contents

Preface: About this guide.....	xi
Who should read this guide?.....	xi
How is this book organized?.....	xi
Typographic Conventions.....	xi
References.....	xii
How do I contact BlueCat Customer Care?.....	xii
 Chapter 1: What's New.....	 13
New API Methods.....	14
New Constants.....	14
Changes in 8.3.0 API.....	14
 Chapter 2: Overview.....	 17
Manipulating Address Manager Objects.....	18
Implementation.....	18
Finding Objects.....	18
Address Manager Object Hierarchy.....	18
Logging In and Out of Address Manager.....	20
Session Management.....	21
Security.....	21
Enabling SSL in Perl Clients.....	21
Enabling SSL in Java Clients.....	21
 Chapter 3: The Address Manager API.....	 23
Web Services API.....	24
SOAP Binding Address.....	24
SOAP Ports.....	24
Maintaining state with cookies.....	24
API Objects.....	24
APIEntity Class.....	25
APIAccessRight Class.....	25
APIDeploymentRole Class.....	25
APIDeploymentOption Class.....	26
APIUserDefinedField Class.....	26
ResponsePolicySearchResult Class.....	27
API Sessions.....	27
Log in and Log out.....	27
System Information.....	27
Working with Java API.....	28
Connecting to Address Manager.....	28
Logging in and out.....	29
Getting Objects.....	30
Adding Objects.....	31
Deleting Objects.....	31
Sequence of Calls in the Client.....	32
Changed API methods for Java users.....	32

- Available Java Classes.....33
- Working with Perl API.....33
 - Connecting to Address Manager.....34
 - Logging in and out.....34
 - Getting, Adding, Deleting, and Updating Objects.....35
- REST API.....35
 - REST vs SOAP.....36
 - Authentication and authorization.....36
 - REST API Examples.....37
 - Limitations.....41
 - REST API troubleshooting.....41

Chapter 4: API Object Methods..... 43

- Generic Methods..... 44
 - Getting Objects..... 44
 - Searching and Retrieving Entities..... 45
 - Updating Objects..... 50
 - Deleting Objects..... 52
 - Linked Entities..... 53
 - Changing Locale.....55
- User-defined Fields..... 56
 - Setting UDF values when adding or updating..... 56
 - Getting User-defined Fields.....57
- IPAM..... 58
 - IPv4 Blocks.....58
 - IPv4 Networks.....61
 - IPv4 Network Templates..... 67
 - IPv4 addresses..... 71
 - Additional IP Addresses..... 76
 - IPv4 Group.....77
 - IPv4 Objects..... 78
 - IPv4 Discovery and Reconciliation.....80
 - IPv6 Objects..... 85
 - Provision Devices..... 91
- DHCP.....93
 - IPv4 DHCP Ranges.....93
 - IPv6 DHCP Ranges.....96
 - DHCP Client Options.....99
 - DHCP6 Client Options.....100
 - DHCP Custom Options.....102
 - DHCP Service Options.....103
 - DHCP6 Service Options.....105
 - DHCP Vendor Profiles and Options.....106
 - DHCP Match Classes.....110
 - DHCP Raw Options.....112
 - Shared Networks.....112
- DNS.....114
 - DNS Views.....114
 - DNS Zones.....115
 - DNS Zone Templates.....118
 - ENUM Zones.....120
 - ENUM Numbers.....121
 - DNS Resource Records.....121
 - DNS Options.....136
 - DNS Raw Option.....139

DNS Response Policies.....	140
Reverse zone name format.....	142
Deployment options.....	143
Getting deployment options.....	143
Raw deployment option.....	144
TFTP.....	145
TFTP Groups.....	145
TFTP Folders.....	146
TFTP Files.....	146
Servers and Deployment.....	147
Servers.....	147
Server Group.....	153
DNS and DHCP Deployment Roles.....	154
DHCP Deployment Roles.....	156
DNS Deployment Roles.....	157
TFTP Deployment Roles.....	160
Crossover High Availability (xHA).....	160
Requirements for creating an xHA pair.....	161
Creating an xHA.....	161
Breaking an xHA.....	164
xHA Failover.....	165
Address Manager Objects.....	165
Configurations.....	165
Groups and Users.....	167
Authenticators.....	169
Access Rights.....	170
Object Tag Groups.....	173
Object Tags.....	174
Locations.....	175
Database Management.....	176
Devices.....	178
MAC Pools.....	179
MAC Addresses.....	181
Workflow Change Requests.....	182
Migration.....	183
Migrate a File.....	183
Migration Status.....	183
Collecting Data.....	184
Start Probe.....	184
Get Probe Status.....	184
Get Probe Data.....	184

Chapter 5: API Constants..... 187

Access Right Values.....	189
Additional IP Service Type.....	189
Configuration Setting.....	189
Deployment Services.....	189
Deployment Status.....	189
Device Properties.....	190
DHCP Class Match Criteria.....	191
DHCP Client Options.....	191
DHCP6 Client Options.....	194
DHCP Custom Option Types.....	194
DHCP Deployment Role Types.....	195
DHCP Service Options.....	195

DHCPServiceOptionConstants.....	196
DHCP6 Service Options.....	196
DNS Deployment Role Type.....	197
DNS Options.....	197
DNS Option Values.....	198
DNSSEC Key Format.....	199
DNS Zones Deployment Validation Check.....	199
Entity Categories.....	199
ENUM Services.....	200
IP Assignment Action Values.....	201
IP Discovery Type.....	201
Object Properties.....	201
Object Types.....	209
Option Types.....	211
PositionRangeBy.....	212
Response Policy Type.....	212
Response Policy Search Scopes.....	212
Reverse Zone Format Type.....	212
Server Capability Profiles.....	213
Service Types.....	213
SNMP Version.....	213
SNMP Security Levels.....	214
SNMP Authentication Type.....	214
SNMP Privacy Type.....	214
Traversal Methodology.....	214
User Access Type.....	215
User-defined Field Type.....	215
User-defined Field Validator Properties.....	215
User History Privileges.....	215
User Security Privileges.....	216
User Type.....	216
Vendor Profile Option Types.....	216
Workflow Levels.....	217
Defined Probe Values.....	217
Probe Status Values.....	217
Chapter 6: API Method Reference.....	219
API Sessions.....	220
Generic Methods.....	220
Linked Entities.....	220
Changing Locale.....	221
User-defined Fields.....	221
IPAM.....	221
IPv4 Blocks.....	221
IPv4 Networks.....	222
IPv4 Network Templates.....	223
IPv4 Addresses.....	223
IPv4 Objects.....	223
IPv4 Group.....	224
IPv4 Discovery and Reconciliation.....	224
IPv6 Objects.....	224
Provision Devices.....	225
DHCP.....	225
IPv4 DHCP Ranges.....	225
IPv6 DHCP Ranges.....	226

DHCP Client Options.....	226
DHCP6 Client Options.....	226
DHCP Custom Options.....	227
DHCP Service Options.....	227
DHCP6 Service Options.....	227
DHCP Vendor Options.....	227
DHCP Match Classes.....	228
Shared Networks.....	228
DNS.....	228
DNS Views.....	228
DNS Zones.....	229
DNS Zone Templates.....	229
ENUM Zones.....	229
ENUM Numbers.....	229
Generic Resource Records.....	230
NAPTR Records.....	230
External Host Records.....	230
Host Records.....	230
Alias Records.....	231
Text Records.....	231
HINFO Records.....	231
MX Records.....	231
SRV Records.....	232
Start of Authority Records.....	232
Generic Records.....	232
DNS Options.....	232
DNS Response Policies.....	233
Reverse Zone Name Format.....	233
Deployment Options.....	233
TFTP.....	233
TFTP Groups.....	233
TFTP Folders.....	234
TFTP Files.....	234
Servers and Deployment.....	234
Servers.....	234
Server Group.....	235
DNS and DHCP Deployment Roles.....	235
DHCP Deployment Roles.....	235
DNS Deployment Roles.....	236
TFTP Deployment Roles.....	236
Crossover High Availability (XHA).....	236
Address Manager Objects.....	236
Configurations.....	236
Groups and Users.....	237
Authenticators.....	237
Access Rights.....	237
Devices.....	238
Object Tag Groups.....	238
Object Tags.....	238
Locations.....	238
Database Management.....	239
MAC Pools.....	239
MAC Addresses.....	239
Workflow Change Requests.....	239
Migration.....	240
Collecting Data.....	240

Chapter 7: Property Options Reference.....241

- Property Options.....242
 - Configuration.....242
 - Views and Zones.....242
 - Resource Records.....242
 - Admin.....244
 - Tags.....244
 - Vendor Profiles.....244
 - DNSSEC.....245
 - TFTP Objects.....245
 - MAC Pool Objects.....245
 - Device.....245
 - Location.....246
 - Kerberos Realms.....247
 - Server.....247
 - IPv4Objects.....248
 - IPv6Objects.....251
 - DeploymentRoles.....252
 - Access right.....252
- IP Address States.....252
 - IPv4.....252
 - IPv6.....253

About this guide

The *Address Manager API Guide* describes the Application Programming Interface (API) for controlling Address Manager IP Address Management (IPAM) appliances and virtual machines and offers instructions on its implementation and usage. IP Address Management includes management of DNS (Domain Name Services), DHCP (Dynamic Host Control Protocol), and IP inventories.

Who should read this guide?

This book is intended for a highly technical audience. This audience may include developers, IT planners, and IPAM, DNS, and DHCP administrators.

How is this book organized?

Procedural information is organized in numbered points to help in rapid implementation. Examples that are longer than a couple of lines are separated from other information, and are clearly marked with an Example heading. All examples are in pseudocode unless otherwise indicated.

Overview—describes the general functionality of the Address Manager API. This chapter describes the common sequences of operations and lists the available objects and methods.

The Address Manager API—describes the SOAP, Java, Perl, and REST API implementations.

API Object Methods—describes the methods available for the Address Manager object types.


API Constants—lists the constants used by the Address Manager API methods.





API Method Reference—lists methods by Address Manager object type.

Property Options Reference—lists available properties and IP address states.

Typographic Conventions

This guide uses the following conventions:

Bold	Command line options, button names, fields, tabs, and icons in the user interface. New terms being defined. Dialog box, window, and screen names.
<i>Blue italic</i>	Cross references and hypertext links within the document. Hypertext links to external URLs.
Monospace	Source code examples and terminal output.
<i>Monospace italic</i>	Variables in code examples.
<i>Normal italic</i>	Emphasis within a concept description.
 Attention:	This icon appears alongside an Attention. Attentions usually appear with critical information or with actions that might result in unexpected behavior.

	Caution:	This icon appears alongside a Caution. Cautions usually appear where performing an action may be dangerous to the user or to the equipment, or where data may be corrupted or incomplete if the caution is not observed.
	Note:	This icon appears alongside a Note. Notes give additional details on the expected behavior about the material presented in concepts and procedures.
	Tip:	This icon appears alongside a Tip. Tips are similar to Notes and suggest alternative ways to accomplish a task or provide ideas for using the product in the most effective way.
	Warning:	This icon appears alongside a Warning. Warnings are more severe than Cautions in that they alert you to actions that could result in potential data loss or a service outage.

References

Working with an IPAM system requires in-depth knowledge of many subject areas, including DNS, DHCP, IP Inventory Management and General Networking.

The following references are provided for readers who require more background knowledge before working with Address Manager.

- The *DHCP Handbook* by Ralph Droms and Ted Lemon, SAMS Publishing, ISBN 0-672-32327-3
- *Pro DNS and BIND* by Ron Aitchison, Apress, ISBN 1-59059-494-0
- The Internet System Consortium website (www.isc.org). This site also hosts the BIND FAQ at www.isc.org/sw/bind and the DHCP FAQ at www.isc.org/software/dhcp.

How do I contact BlueCat Customer Care?

For 24/7/365 support, visit the BlueCat Customer CARE Portal at <https://care.bluecatnetworks.com>.

What's New

Topics:

- [New API Methods](#)
- [New Constants](#)
- [Changes in 8.3.0 API](#)

This chapter provides brief lists of new methods and updates to the existing API methods.

New API Methods

The Address Manager API includes the following new methods:

Add DHCPv6 Range by Size

Use `addDHCP6RangeBySize` to add a new IPv6 DHCP range by size. For more information, refer to [Add IPv6 DHCP Range By Size](#) on page 97.

Update User Password

Use `updateUserPassword` to update Address Manager user password. For more information, refer to [Update User Password](#) on page 169.

Update Address Manager System Password

Use `void updateBAMSystemUserPassword` to update Address Manager system user password. For more information, refer to [8966](#).

Update DNS/DHCP Server System Password

Use `void updateBDDSSystemUserPassword` to update Address Manager system user password. For more information, refer to [8966](#).

New Constants

The Address Manager API includes the following new constants:

New Constants in v8.3.0

- New constant **SLAVE_ZONE_NOTIFICATIONS** has been added to the DNS deployment option when using the following methods:
 - `getDeploymentOptions`
 - `addDNSDeploymentOption`
 - `getDNSDeploymentOption`
 - `updateDNSDeploymentOption`
 - `deleteDNSDeploymentOption`

The following rules apply when using the new **SLAVE_ZONE_NOTIFICATIONS** DNS option:

- This option can only be added at Configuration, DNS view and DNS zone levels.
- You must associate this option with a single server. This option cannot be associated with a server group object.
- This option can only be associated with one server under the same level.
- The value of the option can be either *enable* or *disable*.

Changes in 8.3.0 API

Includes the following changes in behavior:

Added support for CAA resource records

The `addGenericRecord` API method has been updated to support the Certificate Authority Authorization (CAA) generic resource record type.

Resolved issues

An error related to the same FQDN in multiple resource records has been resolved.

Issue—Previously, when using the following API methods, if there is same FQDN assigned to a normal resource record and an external host record, an error was triggered:

- `addAliasRecord()`
- `addSRVRecord()`
- `addMXRecord()`
- `addResourceRecord()` for CNAME, SRV and MX

Resolved—In Address Manager v8.3.0, when there are normal resource records such as Host and CNAME and external host records using the same FQDN, the normal resource records will take precedence over external host records and will be used to create link records such as CNAME, SRV and MX.



Note: When there are multiple normal resource records with the same FQDN an error will be triggered.

Overview

Topics:

- [*Manipulating Address Manager Objects*](#)
- [*Implementation*](#)
- [*Finding Objects*](#)
- [*Address Manager Object Hierarchy*](#)
- [*Logging In and Out of Address Manager*](#)
- [*Session Management*](#)
- [*Security*](#)

The Address Manager API (Application Programming Interface) is a web service that make Address Manager accessible to any system that has standard network or Internet access.

Use SOAP (Simple Object Access Protocol) or REST (Representational state transfer) to access this web service. The web service has a WSDL (Web Service Description Language) file that can be viewed in a browser. Use the WSDL file to generate client artifacts, such as methods and serialized classes. Implementers can use this service directly. Java and Perl API implementations are also provided.

Manipulating Address Manager Objects

Remotely manipulates objects through the Address Manager API, using a combination of generic and type-specific methods.

This manipulation involves adding objects, getting objects by name, updating them, adding new child objects, access rights or object tags to them, and deleting them. The Address Manager API also includes various query functions, such as a check to see if an IP address is allocated. To access the Address Manager API, the script or application must log in to Address Manager as an API user.

Implementation

Sessions in the Address Manager API are implemented as Perl or Java programs, or accessed directly through the web service.

Generally, these sessions log in, perform a function, and then log out again. Provided the script is successful, the next script then logs in, performs its function, and then logs out.

1. Connect to the Address Manager server.
2. Log in.
3. Get the initial configuration, user, group or tag group object and proceed to step 4, or retrieve a specific object by name or ID using **getEntity()** and proceed to step 7.
4. Use **getEntities()** to find the children of the initial object.
5. Use **getEntity()** or a less generic method to select a single entity.
6. Add a child object or affect the current object.
7. Log out.

Variations on this pattern are possible, provided the API implementation can provide sufficient information to retrieve the required objects.

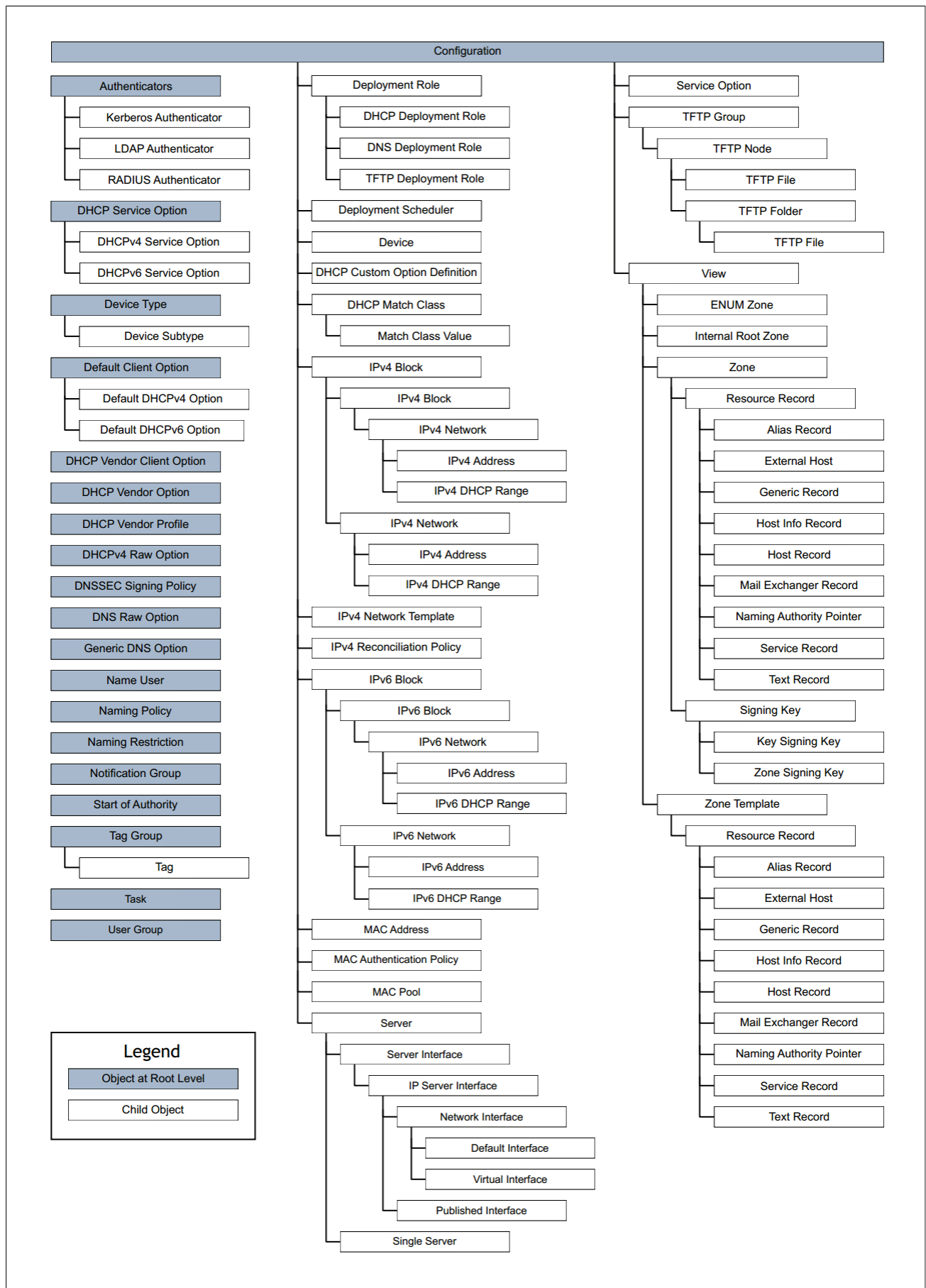
Finding Objects

These search functions provide quick access to Address Manager objects. In most cases this will eliminate the need to search through the entire Address Manager object tree.

- **searchByCategory()** returns an array of entities by searching keywords associated with objects of a specified object category. For more information, refer to [Search by Category](#) on page 48.
- **searchByObjectTypes()** returns an array of entities by searching keywords associated with objects of a specified object type. You can search for multiple object types with a single method call. For more information, refer to [Search by Object Types](#) on page 48.
- **searchResponsePolicyItems** returns an array of entities by searching keywords associated with objects of a specified policy item. For more information, refer to [Search Response Policies](#) on page 141.

Address Manager Object Hierarchy

The web services API is designed to facilitate various types of development, and can be implemented in many different ways. Ultimately, client-side implementations can model the way that data is stored in Address Manager in order to persist objects temporarily. Keeping this structure in mind will help you create caching or reference data structures within client implementations.



Logging In and Out of Address Manager

To access the Address Manager API, the script or application must log in to Address Manager using an API user account.

All API implementations must first connect to the Address Manager API service, and then log in to start a session. None of the Address Manager API functionality described in the following subsections is accessible unless an API user logs on to the system. Address Manager API users have a specific access type (API) and cannot log in to Address Manager through the GUI (Graphical User Interface). Similarly, non-API users cannot connect to Address Manager through the API interface.

To connect to the service, you need the IP address for the Address Manager server. Logging in creates a session for that user with a timeout limit corresponding to that set for all users. API users must also log out after the required operations have been completed.

For more information about API access, refer to the *Address Manager Administration Guide* or the online Help.

1. Using the Address Manager web interface, log in to Address Manager as an administrator.
2. On the *Administration* page, click **Users and Groups**. The *Users and Groups* page appears.
3. In the Users section, click **New**. The *Add User* page appears.
4. In the User section, type a name in the **Username** field.
5. In the Authentication section, type and confirm the API user password in the **Password** and **Confirm Password** fields. If external authenticators are available, an **Other** checkbox and a list of authenticators appears in the Authentication section. To use an external authenticator for the API user, click the **Other** checkbox, and then select an authenticator from the list.
6. In the Extra Information section, set the following parameters:

E-mail Address

Type an email address for the API user. This information is required.

Phone number

Type a phone number for the API user. This information is optional.

7. In the **User Access** section you define the user type, security and history privileges, and access type:

Type of User

Select the type of user, either **Non-Administrator** or **Administrator**. **Non-Administrator** users have access only to DNS and IPAM management functions. **Administrator** users have unlimited access to all Address Manager functions.

Security Privilege

select a security privilege type from the drop-down list. This field is available only for **Non-Administrator** users with **GUI**, **API**, or **GUI and API** access.

History Privilege

select a history privilege type from the drop-down list. This field is available only for **Non-Administrator** users with **GUI**, or **GUI and API** access.

Access Type

select the type of access; **GUI**, **API**, or **GUI and API**. **GUI** (Graphical User Interface) users can access Address Manager only through the Address Manager web interface. **API** (Application Programming Interface) users can access Address Manager only through the API. **GUI and API** users can access Address Manager either through the Address Manager web interface or the API.

8. In the **Assign to Group** section, you can assign the user to one or more existing user groups. In the text field, type the name of a user group. As you type, a list of user groups matching your text appears. Select a name from the list, and then click **Add** to the right of the text field.

9. In the **Change Control** section, add comments to describe the changes. This step is optional but may be set to be required.
10. Click **Add** at the bottom of the page.

Session Management

Web services do not define a standard for session management. Address Manager maintains a session ID to associate it with the Address Manager database session obtained from the initial log in attempt. This accounts for the stateless nature of the HTTP protocol.

Address Manager uses cookies to maintain state. When using WSDL-generated classes, be sure to enable cookies on your system.

Security

The web service endpoint can be made secure by enabling SSL support using the Address Manager Administration Console.

This enables SSL for all Address Manager services, including the web interface. For instructions on enabling SSL on Address Manager, refer to *HTTPS* in the *Address Manager Administration Guide*.

Enabling SSL in Perl Clients

To enable SSL in Perl scripts, you need to install the Crypt-SSLeay package.

You can download the package from <http://search.cpan.org/dist/Crypt-SSLeay/>.

Perl scripts can use SSL in their web service calls by using *https* instead of *http* in the proxy definition.

➔ **Note:** To ensure that API scripts will continue to operate under the higher security implemented in Address Manager v3.7.1 and later, the following conditions must be met:

- Perl version must be v5.12 or later.
- When enabling SSL, you must access Address Manager using the same host name that is defined by the SSL certificate, not the IP address or other canonical names. If you cannot use that host name, you need to add the following line to the beginning of the script to avoid the API call failure due to the difference in host name as part of request and certificate.

```
$ENV{'PERL_LWP_SSL_VERIFY_HOSTNAME'} = 0;
```

➔ **Note:** This line will disable the validation of SSL certifications for the context of the execution.

Enabling SSL in Java Clients

Steps to enable SSL for Java clients.

Follow these Before beginning, ensure that the Java Development Kit is installed on the client workstation.

1. On Address Manager, enable HTTPS support. For instructions on enabling SSL on Address Manager, refer to *HTTPS* in *Chapter 10: Appliance Settings* in the *Address Manager Administration Guide*.
2. On the client workstation, create a directory to hold the keystore.
3. On Address Manager, locate the **/data/server/conf/server.cert** file, and then copy it to the keystore directory on your client workstation. If SSH is enabled on Address Manager, use an SSH client to copy the file.
4. On the client workstation, navigate to the keystore directory. Ensure that the directory contains the **server.cert** file.

5. Execute the following command: `javaHomePath/bin/keytool -import -trustcacerts -alias ProteusAPI -file server.cert -keystore client.ks -storepass bluecat`
6. Ensure that a **client.ks** file has been generated and appears in the keystore directory.
7. Delete the **server.cert** file from the keystore directory on the client workstation.
8. When connecting to service, use the following command to call the **ProteusAPIUtils.connect()** method: `ProteusAPIUtils.connect(IPAddress, true, pathToClientKeystore\client.ks)`; where *IPAddress* is the IP address of the Address Manager server, and *pathToClientKeystore* is the path to the keystore directory on the client workstation.

The Address Manager API

Topics:

- [Web Services API](#)
- [API Objects](#)
- [API Sessions](#)
- [Working with Java API](#)
- [Working with Perl API](#)
- [REST API](#)

This section describes different types of API implementations.

For integrating into different types of network environments, Address Manager includes the following implementations:

- The most open implementation is the Address Manager service-oriented architecture implementing SOAP web services. Any client able to take advantage of web services can use this implementation.
- RESTful API calls to access Address Manager are supported.
- A Java API implementation is provided to integrate Address Manager into *n*-tier Java architectures.
- A Perl API implementation is provided for environments where scripting is preferred.

Web Services API

The Address Manager API is a SOAP web service, so it has an accessible WSDL file.

You can access this file and generate your own classes and methods to use when connecting to the service. To view the WSDL file in a browser, go to **http://AddressManagerAddress/Services/API?wsdl** address. If HTTPS is enabled on Address Manager, use the HTTPS protocol in the address.

SOAP Binding Address

The WSDL file uses the Address Manager server's host-name as the soap:address location.

For a Address Manager appliance with the factory default host-name, the Address Manager API service looks like this example:

```
<service name='ProteusAPI'>
  <port binding='tns:ProteusAPIBinding' name='ProteusAPIPort'>
    <soap:address location='http://new.server/Services/API' />
  </port>
</service>
```

To configure the soap:address location attribute to your Address Manager appliance, download a copy of the WSDL file to your workstation. Edit the local copy of the WSDL file to change the soap:address location to the required address. Configure your SOAP tools to load the WSDL file from your local copy of the file, rather than from the Address Manager appliance.

SOAP Ports

To access the Address Manager API, use the following ports:

- Port 80 when using HTTP
- Port 443 when using HTTPS

Maintaining state with cookies

Address Manager uses cookies to maintain state.

When using WSDL-generated classes, be sure to enable cookies on your system.

API Objects

The web service defines objects representing all Address Manager object types supported in the service.

These objects can be added, retrieved, manipulated, and deleted. For a list of objects and methods, refer to [API Method Reference](#) on page 219.

The following classes reference all objects within the web service:

- APIEntity
- APIAccessRight
- APIDeploymentRole
- APIDeploymentOption
- APIUserDefinedField
- ResponsePolicySearchResult

APIEntity Class

This class represents all entities except options, roles, and access rights. It manages all other types by passing the values for the object as a delimited properties string of name–value pairs.

The properties for each object are listed in [API Object Methods](#) on page 43.

id	the database ID of the object in Address Manager.
name	the field name, which might be null.
type	the class name of the object. For example, a configuration object has a type equal to Configuration . This field cannot be null. A list of types is part of the API client (Java and Perl).
properties	a string that contains properties for the object in <i>attribute=value</i> format, with each separated by a (pipe) character. For example, a host record object may have a properties field such as ttl=123 comments=my comment . This field can be null.

APIAccessRight Class

This class controls Access Rights objects.

entityId	the database ID of the object to which the access right applies. This value must be greater than 0.
userId	the database ID of the owner of the access right. This value must be greater than 0.
value	the default access right (HIDE, VIEW, ADD, CHANGE, or FULL). This field cannot be null.
overrides	indicates the types that are to be overridden in the access right in the format <i>objectType=accessRightValue</i> where <i>objectType</i> is the same object type used in APIEntity and <i>accessRightValue</i> is one of HIDE, VIEW, ADD, CHANGE or FULL. Multiple override elements are separated by a (pipe) character.
properties	a string containing extra properties for the object in the format <i>attribute=value</i> .

APIDeploymentRole Class

Manages the deployment roles that control the services provided by Address Manager-managed servers. These objects support the standard object functions.

id	the database ID of the deployment role in Address Manager.
type	the type of the role (NONE, MASTER, MASTER_HIDDEN, SLAVE, SLAVE_STEALTH, FORWARDER, STUB, RECURSION, PEER, or AD_MASTER.) This field cannot be null.
service	DNS, DHCP, or TFTP. This field cannot be null.
entityId	the database ID of entity. This value must be greater than 0.
serverInterfaceId	the database ID of the server interface. This value must be greater than 0.
properties	a string containing extra properties for the object in the format <i>attribute=value</i> . This field can be null if used for forward space. A ViewId property must be provided to assign DNS Roles to a Network or Block for a particular DNS View (reverse space). Multiple properties are separated by a (pipe) character.

APIDeploymentOption Class

Deployment options configure both DHCP and DNS services on the network. They are available as DHCP client and service options, as well as standard DNS options. Deployment options support the standard object functions.

id	the database ID of the option in Address Manager.
type	the option type listed in Option Types on page 211. This field cannot be null.
name	the name of the option.
value	the single- or multiple-field value of the option; multiple values are comma-separated. This field cannot be null.
properties	a string containing additional properties. This is used for user-defined fields on most objects, but also passes some values that do not have their own specific parameter.

➔ **Note:** When adding the DDNS hostname option, you need to specify the value in the following format: [Type], [Position], [Data] for IP and MAC type, and [Type], [Data] for FIXED type.

Where:

- **Type**—type of DDNS hostname. The possible values are
DHCPServiceOptionConstants.DDNS_HOSTNAME_TYPE_IP,
DHCPServiceOptionConstants.DDNS_HOSTNAME_TYPE_MAC, or
DHCPServiceOptionConstants.DDNS_HOSTNAME_TYPE_FIXED.
- **Position**—specify where you wish to add the data value to the IP or MAC address. The possible values are DHCPServiceOptionConstants.DDNS_HOSTNAME_POSITION_PREPEND, or DHCPServiceOptionConstants.DDNS_HOSTNAME_POSITION_APPEND. This is only required for IP or MAC type with Data.
- **Data**—For IP and MAC address, this value is used to be prepended or appended to the IP address or MAC address. For FIXED type, this value must be specified and will be used for the DDNS hostname. This is optional for IP and MAC type.

APIUserDefinedField Class

User-defined fields can be added to each of the Address Manager object types. This class allows API users to query and gather user-defined fields information for a specified object type.

name	the internal name of the user-defined field.
displayname	the name of the user-defined field that appears to users in the Address Manager interface.
type	the type of the user-defined field. Types are available as constants in the <i>UserDefinedFieldType</i> class. For available constants, refer to User-defined Field Type on page 215.
defaultValue	the default value for the user-defined field.
validatorProperties	the validation properties for the user-defined field. Property names are available as constants in the UserDefinedFieldValidatorProperties class. For available constants, refer to User-defined Field Validator Properties on page 215.
required	the boolean value. If set to true, users must enter data in the field.

hideFromSearch	the boolean value. If set to true, the user-defined field is hidden from the search.
renderAsRadioButton	the boolean value. If set to true, the user-defined field is rendered as a radio button group.

ResponsePolicySearchResult Class

Represents the Response Policy items that are configured either in local Response Policies or BlueCat Security feed data.

name	the name of the Response Policy item.
parentIds	comma-separated values of the parent Response Policy or RP Zone object(s) ID. If policy item is associated with a Response Policy, it is the Response Policy object ID. If policy item is associated with BlueCat Security feed data, it is the RP Zone object ID.
category	the name of the BlueCat Security feed category associated with the policy item. For example, Malicious, Spam or Botnet C&C. For local response policy items, this will be null.
policyType	the type of response policy. For example, whitelist, blacklist, redirect or blackhole.
configId	the object ID of the parent configuration in which the Response Policy item is configured.

API Sessions

Address Manager API session methods control the connection, log in, and log out processes. There is also a method to return system information about the appliance.

Log in and Log out

Log in and log out of the Address Manager system.

You must use an API user account to access the Address Manager API.

To log in, use the following method, passing the API user name and password:

```
login( String name, String password )
```

To log out, use the following method:

```
logout();
```

System Information

Retrieves Address Manager system information through the API.

To retrieve system information, use `String getSystemInfo()`. This method returns system information in the following format:

```
hostName=value|version=value|address=value|clusterRole=value|
replicationRole=value|replicationStatus=value|entityCount=value|
databaseSize=value|loggedInUsers=value
```

Parameter:

hostName	The host name of the Address Manager server.
version	The version of the Address Manager software.
address	The IP address of the Address Manager server.
clusterRole	The role of the server in an XHA pair, either primary or secondary.
replicationRole	The role of the server in database replication, either primary or secondary.
replicationStatus	The status of the replication service on the Address Manager server.
entityCount	The number of entities within the Address Manager database.
databaseSize	The size, in megabytes, of the Address Manager database.
loggedInUsers	The number of users presently logged in to Address Manager.

Working with Java API

Implement the Address Manager API in Java.

To use the Address Manager API with Java, you may need to install the Metro implementation of SOAP. Down package metro_2.3.1 from <https://metro.java.net/2.3.1/>.

To execute the Address Manager API in Java, you need the following two jar files:

- api.jar
- commons-logging-1.1.3.jar: Apache Commons Logging for logging



Note: Java environment support

api.jar can be used in Java 7 and Java 8 environment to communicate with Address Manager.

Connecting to Address Manager

To begin an API session, you must first connect to the Address Manager server. The following methods are available in *APILoginUtils.java* to establish the connection:

HTTP session

```
ProteusAPI connect ( String hostIP )
```

Parameter	Description
hostIP	The addresses of Address Manager appliance.

HTTPS session

```
ProteusAPI secureConnect ( String hostIP, String trustStoreLocation,  
String passphrase )
```

Parameter	Description
hostIP	The addresses of Address Manager appliance.
trustStoreLocation	The location of the certificate on the client.
passphrase	The passphrase of the certificate on the client.

Output / Response

This method returns a **ProteusAPI** reference containing the methods for the API.

Deprecated API session method

The following method in *ProteusAPIUtils.java* has been deprecated.

```
ProteusAPI_PortType connect ( String address, boolean enableSSL,
    String keystoreLocation )
```

BlueCat recommends using `ProteusAPI connect()` and `ProteusAPI secureConnect()` methods instead.

Parameter	Description
address	The addresses of Address Manager appliance.
enableSSL	If security is enabled on the Address Manager appliance, set this flag to true.
keystoreLocation	The location of the certificate on the client.

➔ **Note:** If you are using HTTP, address is the only required parameter for this method.

Output/Response

This method returns a **ProteusAPI_PortType** reference containing the methods for the API.

Example

Connect to the service using the `connect(address)` function from **APILoginUtils**.

Refer to the following example:

```
// connect to Address Manager
ProteusAPI service = APILoginUtils.connect( "HOST" );
// connect to Address Manager securely
ProteusAPI service = APILoginUtils.secureConnect( "HOST", "KEYSTORE_PATH",
    "KEYSTORE_PASSPHASE" );
```

Deprecated ProteusAPI_PortType class

The use of the **ProteusAPI_PortType** class has been deprecated. Use **ProteusAPI** instead.

Deprecated **ProteusAPI_PortType** class example:

```
ProteusAPI_PortType service = ProteusAPIUtils.connect( "HOST" );
// connect to Address Manager securely
ProteusAPI_PortType service = ProteusAPIUtils.connect( HOST, true,
    "KEYSTORE_PATH" );
```

Logging in and out

- Log in as an API user. Use the **login()** method to log in:

```
// log in and establish a session
service.login( "USERNAME", "PASSWORD" );
```

- After completing API tasks, the API user must log out.

```
service.logout();
```



Note: While a session will expire based on the **Session Timeout** value set on the *Configure Global Settings* page of the Address Manager web interface, an explicit logout is strongly recommended to close the API user session.

Getting Objects

Address Manager provides various methods for getting existing objects. This section provides samples showing a few ways to fetch the objects from Address Manager.

getEntity

There are some variations of `getEntity` such as `getEntityById` and `getEntities`. These methods are useful to find an object. The following examples describe fetching objects under a parent object. In these particular examples, we are finding a configuration in Address Manager.

```
// Signature: getEntityByName( long parentId, String name, String type )
APIEntity existingConfiguration = service.getEntityByName(
    0, // Please refer the tip at the end of this section
    "configName", // The name of the configuration object to be searched
    ObjectTypes.Configuration // Type of object, defined in ObjectTypes interface
);
```

getUserDefinedFields

Use the `getUserDefinedFields()` method to get the list of all user-defined fields of an Object Type. In this example, we use `IP4Block`.

```
// Signature: getUserDefinedFields( String type, boolean
// requiredFieldsOnly )
APIUserDefinedFieldArray fields =
    service.getUserDefinedFields( ObjectTypes.IP4Block, true );

// Get the fields as a list
List <APIUserDefinedField> udfs = fields.getItem();
```

Find objects by hint

Address Manager supports a few methods such as `getZonesByHint` and `getIP4NetworksByHint` to find objects using a hint pattern.

```
// Signature: getZonesByHint( long containerId, int start, int count, String options )

// In this example, we are fetching up to 10 objects with no criteria
APIEntityArray entityArray = service.getZonesByHint( containerId, 0, 10, "" );

// Get the entities as a list
List<APIEntity> entities = entityArray.getItem();

// Fetching 10 objects starting from the 5th element, starting with name 'example'
String options = ObjectProperties.hint + "=example";
entityArray = service.getZonesByHint( containerId, 5, 10, options );
```



Tip: For almost all object types, the `add()` and `get()` methods require *parentID*, which is the ID of the parent object. The following objects can take 0 (zero) as the *parentID*: Configuration, TagGroup, User, UserGroup, and Authenticator.

Adding Objects

There are several ways to add objects to Address Manager via the API. This section contains examples of these methods, including a generic `addEntity` method as well methods for adding specific objects in Address Manager.

Generic `addEntity()`

`addEntity()` is a generic adding method that can be used to create any object. In this example, we are adding a `IP4Block` in Address Manager.

```
// Signature: addEntity( long parentId, APIEntity entity )

// Form the entity object with the data
APIEntity blockEntity = new APIEntity();
// Specify Name
blockEntity.setName( "blockName" );
// Specify Type
blockEntity.setType( ObjectTypes.IP4Block );
// Specify additional properties required to define object
blockEntity.setProperties( ObjectProperties.CIDR + "=192.168.0.0/16" );

// Add this entity under the defined parent
long blockId = service.addEntity( configId, blockEntity );
```

Specific add methods

There are other add methods that can be used to add specific different objects. In this example, we are adding a `IP4Block` in Address Manager.

```
// Signature: addIP4BlockByCIDR( long parentId, String CIDR, String properties )

// Add the block specifying the CIDR. Name has to be specified using properties in this case
String properties = ObjectProperties.name + "=blockName2";
long blockId = service.addIP4BlockByCIDR( configId, "20.0.0/16", properties );
```

Deleting Objects

To delete an object, you can invoke the `delete()` method or one of its variants.

```
// Signature: delete( long objectId )

// Delete API needs object id, which can be retrieved from the fetched
// object
APIEntity entity = service.getEntityByName( configId, "ip4BlockName",
    ObjectTypes.IP4Block );
long objId = entity.getId();

// API call to delete the object
service.delete( objId );
```

Sequence of Calls in the Client

This example adds a host record to an existing zone. It demonstrates a complete session using the Address Manager API from Java. This example implements the following steps:

1. Connect to the Address Manager API service.
2. Log in.
3. Get the parent configuration object by name.
4. Get the parent view object by name.
5. Get the parent zone object by name (you can use the absolute name), and then retrieve its child. In this case, to retrieve example.net, we retrieve the parent **com** to find its child object **example.com**.
6. Define a host record object and add it to the parent zone object.
7. Log out.

```
import com.bluecatnetworks.proteus.api.client.java.APILoginUtils;
import com.bluecatnetworks.proteus.api.client.java.constants.ObjectTypes;
import com.bluecatnetworks.proteus.api.client.java.proxy.APIEntity;
import com.bluecatnetworks.proteus.api.client.java.proxy.ProteusAPI;

public class ProteusAddHostRecord
{
    public static void main( String[] args ) throws Exception
    {
        ProteusAPI service = APILoginUtils.connect( "ProteusIPAddress" );
        service.login( "api_user", "password" );
        APIEntity existingConfiguration = service.getEntityByName( 0,
            "Existing Config", ObjectTypes.Configuration );
        APIEntity existingView =
            service.getEntityByName( existingConfiguration.getId(), "Existing View",
                ObjectTypes.View );
        long hostRecordId = service.addHostRecord( existingView.getId(),
            "www.example.com", "10.0.0.6,10.0.0.8", 1, "" );
        service.logout();
    }
}
```

Changed API methods for Java users

This section lists changed API methods for Java users.

Address Manager API methods will no longer return or accept an array element such as *APIEntity[]* and *APIDeploymentOption[]*.

An array element has been replaced by an array object. Refer to the following table for the list of array elements replaced:

Array element	Replaced array object
APIEntity[]	APIEntityArray
APIAccessRight[]	APIAccessRightArray
APIDeploymentOption[]	APIDeploymentOptionArray
APIDeploymentRole[]	APIDeploymentRoleArray
APIUserDefinedField[]	APIUserDefinedFieldArray
long[]	com.bluecatnetworks.proteus.api.client.java.proxy.LongArray
String[]	com.bluecatnetworks.proteus.api.client.java.proxy.StringArray

Input type example

An array element ([]) input type has been changed to an array object. For example, **long[]** can be replaced with **LongArray**. Refer to the following code example:

```
// Create array[] element containing actual values that need to be send in
// request
Long[] subBlockIds = { subblock1Id, subblock2Id };
// Create new Array object
LongArray subBlocks = new LongArray();
// Add values in array element in Array object
subBlocks.getItem().addAll( Arrays.asList( subBlockIds ) );
// send Array object as part of request.
service.mergeBlocksWithParent( subBlocks );
```

Return type example

An array element ([]) return type has been changed to an array object. For example, **APIEntity[]** is replaced with **APIEntityArray**. Refer to the following code example:

```
// Get the Array object from the response
APIEntityArray apiEntityArray = service.getEntities( configId,
    ObjectTypes.IP4Block, 0, 10 );
// Get actual values in Array object as list
List<APIEntity> apiEntities = apiEntityArray.getItem();
// Iterate through above list to get APIEntity object
for( APIEntity apiEntity : apiEntities )
{
    printEntity( apiEntity );
}
```

Available Java Classes

The API includes a number of classes to facilitate the use of the methods (for example, generating the properties strings).

These classes are discussed below, with the exception of `UserSecurityPrivileges.java` and `UserHistoryPrivileges.java`.

Working with Perl API

The Address Manager API can be implemented in Perl. The module containing the full Perl API implementation is called *API.pm*.

On the workstation running Perl, locate the **lib** directory of your Perl installation and create a new directory, for example, **bam**. Copy the *API.pm* file to the **lib/bam** directory.

➔ **Note:** Older versions of the SOAP::Lite module for Perl may create some warnings. If this is an issue, upgrade to the latest module

All Address Manager methods that take arguments need to use the SOAP::Data package to convert these argument into SOAP compatible arguments. For example, to use the **login()** method, the username argument should look like this:

```
SOAP::Data->name( 'username' )->
    value( $username )->
    type( 'string' )->
    attr( {xmlns => ''} )
```

Where:**username**

The name of the argument (as described by the WSDL).

value

The value to be passed.

type

The SOAP type (for example, string, int, long, or base64).

attr

Necessary to make the SOAP message compatible with the service.

Connecting to Address Manager

Connect to the service using the `connect(address)` function from the **BAMConnection** package.

Refer to the following example:

```
## connect to Address Manager
$service = BAMConnection->connect( "address" => 'ipAddress' );
# use "enableSSL" flag if using SSL
# $service = BAMConnection->connect("address" => 'ipAddress', "enableSSL" =>
'true' );
```

Deprecated the Service package

The use of the **Service** package has been deprecated. Use the **BAMConnection** package instead.

Deprecated **Service** package example:

```
## connect to Address Manager
$service = Service->connect( "address" => 'ipAddress' );
# use "enableSSL" flag if using SSL
# $service = Service->connect("address" => 'ipAddress', "enableSSL" =>
'true' );
```

Logging in and out

- Log in as an API user. Use the **login()** method to log in:

```
## log in and establish a session
$service->login(
  SOAP::Data->name('username')->
    value('apiUserName')->
    type('string')->
    attr({xmlns => ''}),
  SOAP::Data->name('password')->
    value('apiUserPassword')->
    type('string')->
    attr({xmlns => ''}) );
```

- After completing API tasks, the API user must log out.

```
$service->logout();
```



Note: While a session will expire based on the **Session Timeout** value set on the *Configure Global Settings* page of the Address Manager web interface, an explicit logout is strongly recommended to close the API user session.

Getting, Adding, Deleting, and Updating Objects

- Use the **get()**, **add()**, **delete()**, and **update()** methods to manipulate Address Manager entities. This example shows the addition of a new configuration with a shared network:

```
# Add a new configuration with a shared network
my $configuration = APIEntity->new( "id" => 0,
    "name" => "Test Configuration",
    "type" => ObjectTypes::Configuration,
    "properties" =>
        ObjectProperties::sharedNetwork."=".$existingSharedNetwork1->
            get_id()."|" );
my $configurationId = $service->addEntity( SOAP::Data->type( 'long' )->
    name( 'parentId' )->
    value( 0 )->
    attr({xmlns => ''}),
    SOAP::Data->type( 'APIEntity' )->name( 'entity' )->
    value( $configuration )->
    attr({xmlns => ''}) )->
    result;
print "New Configuration id = ".$configurationId->get_id()."\n";
```

- Use the **getUserDefinedFields()** method to find the user-defined fields with their settings and values in Address Manager. For example:

```
my @udfs= $service->getUserDefinedFields( SOAP::Data->type( 'string' )->
    name( 'type' )->value( ObjectTypes::Device )->attr({xmlns => ''}),
    SOAP::Data->type( 'boolean' )->name( 'requiredFieldsOnly' )->
    value( 'false' )
    ->attr({xmlns => ''}) )
->valueof('//getUserDefinedFieldsResponse/return/item');

print "number of fields=" . @udfs . "\n";

for my $eachUDF ( @udfs )
{
    my $udf = BAMConnection->blessAPIUserDefinedField( "object" => $eachUDF );
    print "Object-----\n";
    print $udf->get_name() . "\n";
    print "Name=" . $udf->get_name() . "\n";
    print "DisplayName=" . $udf->get_displayName() . "\n";
    print "Type=" . $udf->get_type() . "\n";
    print "defaultValue=" . $udf->get_defaultValue() . "\n";
    print "Validator Properties=" . $udf->get_validatorProperties() . "\n";
    print "PredefinedValues=" . $udf->get_predefinedValues() . "\n";
    print "Required=" . $udf->get_required() . "\n";
    print "Hide from search=" . $udf->get_hideFromSearch() . "\n";
    print "Radio=" . $udf->get_renderAsRadioButton() . "\n";
}
```

For almost all object types, the **add()** and most **get()** methods require *parentID*, which is the ID of the parent object. The following objects can take 0 (zero) as the *parentID*: Configuration, TagGroup, User, UserGroup, and Authenticator.

REST API

Address Manager now supports REST APIs to access Address Manager along with conventional SOAP APIs.

This section describes BlueCat's RESTful API implementation such as the invoking format, how to authenticate and authorize, and REST API samples and limitations.

- BlueCat REST APIs support communication over HTTP and HTTPS.
- JSON format supported for sending and receiving data.

REST API format

You can access all BlueCat REST APIs by appending known methods to the base URL `http://<AddressManager_IP or hostname>/Services/REST/v1/<api_method_name>`

For example, to invoke the `getSystemInfo` method, you need to enter:

```
http://192.168.1.2/Services/REST/v1/getSystemInfo
```

Web Application Description Language (WADL)

The Address Manager REST API has an accessible WADL. You can access the WADL file by using `http://<AddressManager_IP or hostname>/Services/REST/application.wadl`

You can use the WADL file to check various parameters of the API such as:

- API signatures
- HTTP methods (GET/POST/PUT/DELETE) for specific calls
- API request parameters and datatype
- Response datatype and media types

REST vs SOAP

REST APIs have many similarities with the widely used SOAP-based APIs supported by Address Manager. However, there are a few differences between REST interface and existing SOAP implementation:

- The names of API methods in REST remain the same as that of SOAP APIs.
- Signatures of all methods including input and output parameters in REST are the same as in SOAP.
- In REST API, various primitive request parameters such as int, long and String are expected as URL query parameters. Whereas in SOAP, all the request parameters are communicated as part of XML body.
- Complex parameter types such as **APIEntity** or **APIDeploymentOption** need to be passed as a part of HTTP body of the call in JSON format.

Authentication and authorization

Address Manager uses a token-based authentication and authorization. Once generated, the token must be used when invoking every subsequent API method in Address Manager.

Generating a token

You need to generate the authentication and authorization token by invoking the `login` API method. You must use an API user account to access the Address Manager API. If the credentials are invalid, it will fail with an error. For more information about log in and log out methods, refer to [Log in and Log out](#) on page 27.

Log in format:

```
http://<AddressManager_IP or hostname>/Services/REST/v1/login?
username=<username>&password=<password>
```

Output / Response

The body of the JSON response for the above API method will be in the "Session Token-> \${ACTUAL_TOKEN} <- for User : \${USERNAME_PASSED}" and it can be used to extract the authorization token.

The ACTUAL_TOKEN is comprised of the keyword *BAMAuthToken* and a dynamically generated token hash. For example, "BAMAuthToken: 4bippMTQ1ODAzNzgWnJE0MzphcGk=".

Authorizing API methods

The generated authorization token must be passed when invoking any API method. The token needs to be passed as an Authorization property in the request header.

Configuring the token timeout

By default, the generated token expires after 5 minutes. You can change this behavior according to your use case.



Note: For details on how to change the token timeout, refer to knowledge base article [7724](#) on BlueCat Customer Care.

REST API Examples

This section describes how to use REST APIs with main generic method examples.

The four main generic Address Manager APIs use these HTTP RESTful methods in the following examples:

Generic Address Manager API	RESTful HTTP method
get()	GET
update()	PUT
addEntity()	POST
delete()	DELETE

GET request example

All *get* related API methods use the GET method in REST API. In this example, **getDNSDeploymentOption** will be used to retrieve the DNS option.

API call example:

```
http://<AddressManager_IP>/Services/REST/v1/getDNSDeploymentOption?
entityId=101041&name=allow-ddns&serverId=100907
```

URL path:

- /getDNSDeploymentOption

Parameters:

Parameter	Description
entityId	The object ID for the entity to which this deployment option is assigned.
name	The name of the DNS option.
serverId	Specifies the server to which this option is assigned. To retrieve an option that has not been assigned to a server role, set this value to 0(zero).

HTTP Header:

- Authorization: BAMAAuthToken: UTtSjMTQ1ODAzMTgzMDUxMzphcGk=
- Content-Type: application/json

Response

A JSON (as defined in WADL) containing details of the DNS deployment option.

```
{
  "id": 100979,
  "type": "DNS",
  "name": "allow-ddns",
  "value": "any",
  "properties": "inherited=false|"
}
```

Passing String[]

String[] parameter needs to be passed as a repeated URL parameter.

API call example:

```
http://<AddressManager_IP>/Services/REST/customSearch?
filters=filter1=abc&filters=filter2=def&type=IP4Block&options=&start=0&count=10
```

HTTP Header:

- Authorization: BAMAAuthToken: UTtSjMTQ1ODAzMTgzMDUxMzphcGk=
- Content-Type: application/json

HTTP Body:

- None.

POST request example

All *add* operation related API methods use the POST method in REST API. In this example, **addEntity** will be used to add a View object.

API call example:

```
http://<AddressManager_IP>/Services/REST/v1/addEntity?parentId=100936
```

URL path:

- /addEntity

Parameters:

Parameter	Description
parentId	For configurations, always set the parentId value to 0 (zero) , which is the root element.
entity	The configuration object, including its name, sharedNetwork, and user-defined fields (entity will be passed in HTTP Body as JSON)

HTTP Header:

- Authorization: BAMAAuthToken: UTtSjMTQ1ODAzMTgzMDUxMzphcGk=
- Content-Type: application/json

HTTP Body:

- JSON containing information for adding a view.

```
{
  "id":0,
  "name":"testView",
  "type":"View",
  "properties":""
}
```

Response

Returns the character sequence representing the object ID of the new entity.

```
100936
```

Passing long[]

long[] parameter will be passed in the HTTP body in an array representation.

API call example:

```
void mergeBlocksWithParent ( long[] blockIDs)
http://<AddressManager_IP>/Services/REST/mergeBlocksWithParent
```

HTTP Header:

- Authorization: BAMAAuthToken: UTtSjMTQ1ODAzMTgzMDUxMzphcGk=
- Content-Type: application/json

HTTP Body:

```
[
  id1, id2, id3
]
e.g.: [
  100922, 100923
]
```

Response

No response. The method has the void return type.

Passing byte[]

long[] parameter will be passed in the HTTP body.

API call example:

```
void uploadResponsePolicyItems(long parentId, byte[] policyItemsData)
http://<AddressManager_IP>/Services/REST/uploadResponsePolicyItems
```

HTTP Header:

- Authorization: BAMAAuthToken: UTtSjMTQ1ODAzMTgzMDUxMzphcGk=
- Content-Type: application/json

HTTP Body:

- Upload file for above listed API's in REST API client.

Response

No response. The method has the void return type.

PUT request example

All *update* related API methods use the PUT method in REST API. In this example, **updateDNSDeploymentRole** will be used to update a specified DNS deployment role.

API call example:

```
http://<AddressManager_IP>/Services/REST/updateDNSDeploymentRole
```

URL path:

- PUT / updateDNSDeploymentRole

Parameters:

Parameter	Description
role	The DNS deployment role object to be updated. It will be passed in the HTTP body.

HTTP Header:

- Authorization: BAMAAuthToken: UTtSjMTQ1ODAzMTgzMDUxMzphcGk=
- Content-Type: application/json

HTTP Body:

- JSON containing the update information of DNS deployment role.

```
{  "id": 101081,
  "entityId": 101041,
  "serverInterfaceId": 100908,
  "type": "NONE",
  "service": "DNS",
  "properties": "readOnly=false|nsRecordTTL=86400|inherited=false|"
```

Response

No response. The method has the void return type.

DELETE request example

All *delete* related API methods use the DELETE method in REST API. In this example, **deleteDHCPClientDeploymentOption** will be used to delete a specified DHCP client option.

API call example:

```
http://<AddressManager_IP>/Services/REST/v1/deleteDHCPClientDeploymentOption?entityId=101226&name=time-offset&serverId=101217
```

URL path:

- DELETE /deleteDHCPClientDeploymentOption

Parameters:

Parameter	Description
entityId	The object ID for the entity from which the deployment option will be deleted.
Name	The name of the DHCPv4 client option to be deleted (Constant listed in BAM API guide)
serverId	The specific server to which this option is deployed. To delete an option that has not been assigned to a server, set this value to 0 (zero).

HTTP Header:

- Authorization: BAMAAuthToken: UTtSjMTQ1ODAzMTgzMDUxMzphcGk=
- Content-Type: application/json

Response

No response. The method has the void return type.

Limitations

The current limitations of the REST API implementation are as follows:

- No API client is available for REST. BlueCat *api.jar/API.pm* files do not support REST APIs.
- *Get* related APIs with no matching input will return a sparsely populated object.
- REST API session timeout can be configured only through *server.properties*. The Address Manager server needs to be restarted to reflect the new timeout configuration.
- REST API methods that require the body for the API call should be formed correctly by a client. If the body is formed incorrectly, the server will give you an error.

REST API troubleshooting**Invalid credentials for login**

When you use invalid credentials, the `login` API call will fail with the error code 500 and the following message will display: *"Authentication Error: Ensure that your username and password are correct."* You need to login again with the right credentials.

Invalid or expired token

When a token is invalid or expired, API calls will fail with the error code 401 Unauthorized and the following message will display: "UNAUTHORIZED USER". You need to generate a new token by logging in again and use the newly generated token to make subsequent API calls.

Unrecognized field error

If the format of the JSON input request is incorrect, it results in "Unrecognized field" error. For example,

```
Unrecognized field "parentId" (class
  com.bluecatnetworks.proteus.api.service.types.APIEntity), not marked as
  ignorable (4
    known properties: "type" , "id" , "properties" , "name" ])
```


API Object Methods

Topics:

- [Generic Methods](#)
- [User-defined Fields](#)
- [IPAM](#)
- [DHCP](#)
- [DNS](#)
- [Deployment options](#)
- [TFTP](#)
- [Servers and Deployment](#)
- [Crossover High Availability \(xHA\)](#)
- [Address Manager Objects](#)
- [Migration](#)
- [Collecting Data](#)

This chapter lists the methods available in the Address Manager API.

Some of the generic methods include implementation examples in Java and Perl; the others are either described in pseudocode or are extended from generic methods through passing field values.



Note: Address Manager API does not validate the user-defined fields with a pre-defined set of values when adding an object even though the Require Value property of the UDFs is set.

Generic Methods

Many of the object types listed below use the **update()**, **delete()**, and **get()** methods.

While some objects may have specific **get()** methods, the generic methods described here are required in many Address Manager API scripts.

Getting Objects

Generic methods for getting entity values.

- Get entities by name
- Get entities by ID
- Get Entities
- Get Parent

Get Entity by Name

Returns objects from the database referenced by their **name** field.

Output / Response

Returns the requested object from the database.

API call:

```
APIEntity getEntityByName( long parentId, String name, String type )
```

Parameter	Description
parentId	The ID of the target object's parent object.
name	The name of the target object.
type	The type of object returned by the method. This string must be one of the constants listed in Object Types on page 209.

Get Entity by ID

Returns objects from the database referenced by their database ID and with its properties fields populated.

Output / Response

Returns the requested object from the database with its properties fields populated. For more information about the available options, refer to [IPv4Objects](#) on page 248 in the [Property Options Reference](#) section.

API call:

```
APIEntity getEntityById ( long id )
```

Parameter	Description
<i>id</i>	The object ID of the target object.

Get Entities

Returns an array of requested child objects for a given *parentId* value. Some objects returned in the array may not have their properties field set. For those objects, you will need to call them individually using the `getEntityById()` method to populate the properties field.



Note:

- Using `getEntities()` to search users will return all users existing in Address Manager. Use `getLinkedEntities()` or `linkEntities()` to search users under a specific user group.
- Using `getEntities()` to query server objects in configurations containing XHA pairs might result in a connection timeout if any of the servers in an XHA pair are not reachable.

Output / Response

Returns an array of the requested objects from the database without their properties fields populated, or returns an empty array.

API call:

```
APIEntity[] getEntities( long parentId, String type, int start, int count )
```

Parameter	Description
parentId	The object ID of the target object's parent object.
type	The type of object returned. This must be one of the constants listed in Object Types on page 209.
start	Indicates where in the list of objects to start returning objects. The list begins at an index of 0.
count	Indicates the maximum number of child objects to return.

Get Parent

Returns the parent entity of a given entity.

Output / Response

Returns the `APIEntity` for the parent entity with its properties fields populated. For more information about the available options, refer to [IPv4Objects](#) on page 248 in the [Property Options Reference](#) section.

API call:

```
APIEntity getParent ( long entityId )
```

Parameter	Description
long entityId	The entity Id.

Searching and Retrieving Entities

Generic methods for searching and retrieving entities.

- Custom Search
- Search by Category
- Search by Object Types
- Get Entities by Name
- Get Entities by Name Using Options
- Get MAC Address

Supported wildcards in the search string:

You can use the following wildcards when invoking a search method. These wildcards are supported only in the `String` parameter:

- **^**—matches the beginning of a string. For example, **^ex** matches **example** but not **text**.
- **\$**—matches the end of string. For example: **ple\$** matches **example** but not **please**.
- *****—matches zero or more characters within a string. For example: **ex*t** matches **exit** and **excellent**.



Note: You cannot use the following characters in the search string:

- , (comma)
- ' (single quotation mark)
- () (parentheses)
- [] (square brackets)
- { } (braces)
- % (percent)
- ? (question mark)
- + (addition/plus sign)

Custom Search



Search for an array of entities by specifying object properties.

Output / Response

Returns an array of APIEntities matching the specified object properties or returns an empty array. The APIEntity will at least contain *Object Type*, *Object ID*, *Object Name*, and *Object Properties*.

API call:

```
APIEntity[] customSearch ( String[] filters, String type, String[] options, int start, int count )
```

Parameter	Description
filters	<p>The list of properties on which the search will be based. The valid format is <i>Field name=value</i>. Refer to Supported object types and fields for details.</p> <p> Note: The field name is case-sensitive.</p> <p>In addition to the fields that are specified in the table, any user-defined fields will also be supported.</p> <p> Note: The valid format for the <i>Date</i> type user-defined field value is DD-MMM-YYYY. You can also use partial formatting. For example, <i>10-Jan-2016</i>, <i>10-Jan</i>, <i>Jan-2016</i> or <i>2016</i>.</p> <p>For more information on passing a String array through RESTful API calls, refer to Passing String[] on page 38.</p>
type	<p>The object type that you wish to search. The type cannot be null or empty string (""). This must be one for the following object types:</p> <ul style="list-style-type: none"> • IP4Block • IP4Network • IP4Addr • GenericRecord • HostRecord • Any other objects with user-defined fields
options	<p>The list of search options specifying the search behavior. Reserved for future use.</p> <p>For more information on passing a String array through RESTful API calls, refer to Passing String[] on page 38.</p>
start	<p>Indicates where in the list of returned objects to start returning objects. The value must be a non-negative value and cannot be null or empty.</p>

Parameter	Description
count	The maximum number of objects to return. The value must be a positive value between 1 and 1000. This value cannot be null or empty.

Supported object types and fields

Object type	Field name=value
IP4Block	<ul style="list-style-type: none"> • inheritDNSRestrictions=Boolean • pingBeforeAssign=Boolean • reverseZoneSigned=Boolean • allowDupHost=Boolean • inheritDefaultDomains=Boolean • highwatermark=Integer • lowwatermark=Integer
IP4Network	<ul style="list-style-type: none"> • inheritDNSRestrictions=Boolean • pingBeforeAssign=Boolean • reverseZoneSigned=Boolean • allowDupHost=Boolean • inheritDefaultDomains=Boolean • portInfo=Text • highwatermark=Integer • lowwatermark=Integer
IP4Addr	<ul style="list-style-type: none"> • routerPortInfo=Text • portInfo=Text • vlanInfo=Text
GenericRecord	<ul style="list-style-type: none"> • comments=Text • ttl=Long • recordType=Text • rdata=Text
HostRecord	<ul style="list-style-type: none"> • comments=Text • ttl=Long

Java client example

- If using **ProteusAPI_PortType**:

```
//Define filters array
String[] filters = new String[] { "udf_Int=10", "udf_Text=textudfvalue",
    "udf_Date=12-Dec-2016", "udf_Boolean=true", "udf_ea=a@a.com",
    "udf_url=http://a.com", "udf_long=12354" };

//customSearch API call
APIEntity[] entityArray = service.customSearch( filters,
    ObjectTypes.IP4Block, new String[] {}, 0, 1000 );
```

- **If using ProteusAPI:**

```
//Define filters array
StringArray filters = new StringArray();
//filters.getItem().add( "udf_name=udf_value" );
filters.getItem().add( "udf_Int=10" );
filters.getItem().add( "udf_Text=textudfvalue" );
filters.getItem().add( "udf_Date=12-Dec-2016" );
filters.getItem().add( "udf_Boolean=true" );
filters.getItem().add( "udf_ea=a@a.com" );
filters.getItem().add( "udf_url=http://a.com" );
filters.getItem().add( "udf_long=12354" );

//customSearch API call
APIEntityArray entityArray = service.customSearch( filters,
    ObjectTypes.IP4Block, new StringArray(), 0, 1000 );
```

Search by Category

Returns an array of entities by searching for keywords associated with objects of a specified object category.

Output / Response

Returns an array of entities matching the keyword text and the category type, or returns an empty array.

API call:

```
APIEntity[] searchByCategory ( String keyword, String category, int start, int count )
```

Parameter	Description
keyword	The search keyword string. This value cannot be null or empty.
category	The entity category to be searched. This must be one of the entity categories listed in Entity Categories on page 199.
start	Indicates where in the list of returned objects to start returning objects. The list begins at an index of 0. This value cannot be null or empty.
count	The maximum number of objects to return. The default value is 10. This value cannot be null or empty.

Search by Object Types

Returns an array of entities by searching for keywords associated with objects of a specified object type. You can search for multiple object types with a single method call.

Output / Response

Returns an array of entities matching the keyword text and the category type, or returns an empty array.

API call:

```
APIEntity[] searchByObjectTypes ( String keyword, String types, int start, int count )
```

Parameter	Description
keyword	The search keyword string. This value cannot be null or empty.
types	The object types for which to search, specified in the following format: "type1[, type2...] ". The object type must be one of the types listed in Object Types on page 209.

Parameter	Description
start	Indicates where in the list of returned objects to start returning objects. The list begins at an index of 0. This value cannot be null or empty.
count	The maximum number of objects to return. The default value is 10. This value cannot be null or empty.

Get Entities by Name

Returns an array of entities that match the specified parent, name, and object type.

Output / Response

Returns an array of entities. The array is empty if there are no matching entities.

API call:

```
APIEntity[] getEntitiesByName ( long parentId, String name, String type, int start, int count )
```

Parameter	Description
parentId	The object ID of the parent object of the entities to be returned.
name	The name of the entity.
types	The type of object to be returned. This value must be one of the object types listed in Object Types on page 209.
start	Indicates where in the list of returned objects to start returning objects. The list begins at an index of 0. This value cannot be null or empty.
count	The maximum number of objects to return. The default value is 10. This value cannot be null or empty.

Get Entities by Name Using Options

Returns an array of entities that match the specified name and object type. Searching behavior can be changed by using the options.

Output / Response

Returns an array of entities. The array is empty if there are no matching entities.

API call:

```
APIEntity[] getEntitiesByNameUsingOptions ( long parentId, String name, String type, int start, int count, String options )
```

Parameter	Description
parentId	The object ID of the parent object of the entities to be returned.
name	The name of the entity.
types	The type of object to be returned. This value must be one of the object types listed in Object Types on page 209.
start	Indicates where in the list of returned objects to start returning objects. The list begins at an index of 0. This value cannot be null or empty.
count	The maximum number of objects to return. The default value is 10. This value cannot be null or empty.

Parameter	Description
options	<p>A string containing options. Currently the only available option is ObjectProperties.ignoreCase. By default, the value is set to false. Setting this option to true will ignore the case-sensitivity used while searching entities by name.</p> <p><code>ObjectProperties.ignoreCase = [true false]</code></p>

Get MAC Address

Returns an APIEntity for a MAC address.

Output / Response

Returns an APIEntity for the MAC address. Returns an empty APIEntity if the MAC address does not exist. The property string of the returned entity should include the MAC address:

```
address=nn-nn-nn-nn-nn-nn |
```

If the MAC address is in a MAC pool, the property string includes the MAC pool information:

```
macPool=macPoolName |
```

API call:

```
APIEntity getMACAddress ( long configurationId, String macAddress )
```

Parameter	Description
configurationId	The object ID of the configuration in which the MAC address is located.
macAddress	The MAC address in the format nnnnnnnnnnnn, nn-nn-nn-nn-nn-nn or nn:nn:nn:nn:nn:nn, where nn is a hexadecimal value.

Updating Objects

Generic methods for updating an object.

Updating an object involves two steps:

1. Building the object or parameter string used to update the object.
2. Performing the update.

Update

Updates entity objects.

API call:

All entity update statements follow this format:

```
void update ( APIEntity entity )
```

Parameter	Description
entity	The actual API entity passed as an entire object that has its mutable values updated.

Modified behavior for User-defined fields in the `update ()` method:

- **Removing existing UDF values**

Commit the `update()` method with empty UDF value. If the UDF parameter is set to mandatory, the method fails as the UDF parameter cannot be empty.

- **Updating UDF values**

Commit the `update()` method with the new UDF value. If you do not want to update the existing value, leave the UDF parameter and its value unchanged.

- If the UDF parameter is set to mandatory and has a default value, committing the `update()` method with an empty UDF value will take the default value.

Update examples

Provides an example how the update method can be used in Java and Perl.

In this example, an existing shared network is passed to a configuration object as a parameter. After the values in the object or properties string have been set, the `update()` method modifies the value in the Address Manager database. Property values can be a string, long, or integer value. Address Manager uses the appropriate method to process the data for that property.

Java example

This example uses Java to return a managed server as an `APIEntity`, get the properties for the server, add a connected property with the value `true`, set the properties for the server, and then update the server.

```
// Get the object, here Server
APIEntity server = service.getEntityByName(config.getId(), serverName,
ObjectTypes.Server);

// Get the current properties & add a new property
EntityProperties props = new EntityProperties(server.getProperties());
props.addProperty(ObjectProperties.connected, "true");

// Set the changed properties on the object & send it to server to update
server.setProperties(props.getPropertiesString());
service.update(server);
```

Perl example

This example uses Perl to update an external host record.

```
my $externalHostRecord = $service->getEntityById( SOAP::Data-
>type( 'long' )->
name( 'id' )->
value( $externalHostRecordId )->
attr({xmlns => ''}) ) ->result;
$externalHostRecord = BAMConnection->blessAPIEntity( "object" =>
$externalHostRecord );
$externalHostRecord->set_name( "external2.host2.com" );
$service->update( SOAP::Data->type( 'APIEntity' )->
name( 'entity' )->
value( $externalHostRecord )-> attr({xmlns => ''}) );
```

Update with Options

Updates objects requiring a certain behavior that is not covered by the regular `update()` method. This method is currently used for CName, MX and SRV records, and the option is only applicable to these types.

Output / Response

None.

API call:

```
void updateWithOptions ( APIEntity entity, String options )
```

Parameter	Description
entity	The actual API entity to be updated.
options	A string containing the update options. Currently, only one option is supported: linkToExternalHost=boolean . If <code>true</code> , update will search for the external host record specified in linkedRecordName even if a host record with the same exists under the same DNS View. If the external host record is not present, it will throw an exception. If <code>false</code> , update will search for the host record specified in linkedRecordName .

Deleting Objects

Generic methods for deleting an object.

There are two generic methods for getting entity values:

- Delete
- Delete with Options

Delete

Deletes an object using the generic `delete()` method.

Output / Response

None.

API call:

Pass the entity ID from the database identifying the object to be deleted.

```
void delete ( long ObjectId )
```

Parameter	Description
ObjectId	The ID for the object to be deleted.

Delete with Options

Deletes objects that have options associated with their removal. This method currently works only with the deletion of dynamic records from the Address Manager database. When deleted, dynamic records present the option of not dynamically deploying to DNS/DHCP Server.

Output / Response

None.

API call:

```
void deleteWithOptions ( long ObjectId, String options )
```

Parameter	Description
ObjectId	The ID for the object to be deleted. This must be the object ID of a resource record.
options	A string containing the following delete options:

Parameter	Description
	<ul style="list-style-type: none"> • noServerUpdate—a Boolean value. This applies to the dynamic resource records. Set to <i>true</i> to update the record only in the Address Manager web interface. The change will not be deployed to the DNS server. The default value is <i>false</i>. • deleteOrphanedIPAddresses—a Boolean value. This applies to the delete operation on Host Records. Set to <i>true</i> to free IP addresses associated with a host record if no other host records are associated with the IP address. The default value is <i>false</i>.

Linked Entities

Generic methods for getting, link or unlink entities.

- Get Linked Entities
- Link Entities
- Unlink Entities

Get Linked Entities


Returns an array of entities containing the entities linked to a specified entity. The array is empty if there are no linked entities.

Output / Response

Returns an array of entities. The array is empty if there are no linked entities.

API call:

```
APIEntity[] getLinkedEntities ( long entityId, String type, int start, int count)
```

Parameter	Description
entityId	The object ID of the entity for which to return linked entities.
type	<p>The type of linked entities which need to be returned. This value must be one of the types listed in Object Types on page 209.</p> <p> Attention:</p> <ul style="list-style-type: none"> • While specifying a resource record as the entityId, if you want to find all the records (CNAME, MX, or SRV records) having links to this record, you can use RecordWithLink for the type parameter. • When specifying a MAC address as the entityId, this method returns the IPv4 address associated with the MAC address. When appropriate, leaseTime and expiryTime information also appears in the returned properties string.
start	Indicates where in the list of returned objects to start returning objects. The list begins at an index of 0. This value cannot be null or empty.
count	The maximum number of objects to return.

Link Entities

Establishes a link between two specified Address Manager entities.

Output / Response

None.

This method works on the following types of objects and links:

Type of entity for entity1Id	Type of entity for entity2Id	Result
Any entity	Tag	Applies the tag linked to the entity.
Tag	Any entity	Applies the entities from the object tag.
MACPool	MACAddress	Applies the MAC address from the MAC pool.
MACAddress	MACPool	Applies the MAC pool from the MAC address.
User	UserGroup	Applies the user group from the user.
UserGroup	User	Applies the user from the user group.
Location	IP or server object	Applies the IP or server object to a location.
IP or server object	Location	Applies the location to an object.
Server Group	Server	Applies a server to a Server Group object.
Server	Server Group	Applies a Server Group to a server object.

API call:

```
void linkEntities ( long entity1Id, long entity2Id, String properties )
```

Parameter	Description
entity1Id	The object ID of the first entity in the pair of linked entities.
entity2Id	The object ID of the second entity in the pair of linked entities.
properties	Adds object properties, including user-defined fields.

Unlink Entities

Removes the link between two specified Address Manager entities.

Output / Response

None.

This method works on the following types of objects and links:

Type of entity for entity1Id	Type of entity for entity2Id	Result
Any entity	Tag	Removes the tag linked to the entity.
Tag	Any entity	Removes the entities from the object tag.
MACPool	MACAddress	Removes the MAC address from the MAC pool.
MACAddress	MACPool	Removes the MAC pool from the MAC address.

Type of entity for entity1Id	Type of entity for entity2Id	Result
User	UserGroup	Removes the user group from the user.
UserGroup	User	Removes the user from the user group.
Location	IP or server object	Removes the location from an object.
IP or server object	Location	Removes the location from an object.
Server Group	Server	Applies a server to a Server Group object.
Server	Server Group	Applies a Server Group to a server object.

➔ **Note:** To obtain the object ID of the Deny MAC pool, use the `getEntityByName()` method.
For example:

```
entity = service.getEntityByName( <parentId>, <name>, "DenyMACPool" );
```

API call:

```
void unlinkEntities ( long entity1Id, long entity2Id, String properties )
```

Parameter	Description
entity1Id	The object ID of the first entity in the pair of linked entities.
entity2Id	The object ID of the second entity in the pair of linked entities.
properties	Adds object properties, including user-defined fields.

Changing Locale

Address Manager v4.1.1 and greater supports Japanese language in order for API users to update and view the UDF display name in Japanese.

Log in with Options

Log in as API user. To change the locale to Japanese, use the following method when logging in to the Address Manager server. Changing Locale only affects the behavior of `getUserDefinedFields()` and `updateBulkUdf()` methods.

Output / Response

None.

API call:

```
void loginWithOptions ( String userName, String password, String options)
```

Parameter	Description
userName	The username for the API user created using the Address Manager user interface.
password	The password for the API user logging into Address Manager.
options	Following option can be used: <ul style="list-style-type: none"> • locale=ja-JA

User-defined Fields

Add user-defined fields to any Address Manager object type.

These fields are available on all of the object adding and editing forms. A user-defined field can include several enforced data types and can be validated against a complex set of criteria. Any reasonable number of user-defined fields can be added to an object type to track data according to your data schema requirements.

Setting UDF values when adding or updating

Existing user-defined fields for objects can be set and updated through API calls.

Values for these fields can be set in the properties parameter where they are noted the same as any other object field. If a value is set or updated for a non-existent user-defined field, an exception is thrown. These are passed as name-value pairs, and multiple user-defined fields are separated by a |(pipe) character. For example, to set values of two UDFs: TextUDF and IntegerUDF with values testTextValue and 1005, the properties string should be passed as: TextUDF=testTextValue|IntegerUDF=1005.

The following examples are the code snippets to add or update zone with the above Scenario:

Java API Examples

Java clients are equipped with a wrapper class EntityProperties which will help in forming the propertiesString by specifying various properties forming the name value.

Add in Java

```
long parentZoneId; //Id of the parent Zone
EntityProperties properties = new EntityProperties();
properties.addProperty("TextUDF", "testTextValue");
properties.addProperty("IntegerUDF", "1005");
String propertyString = properties.getPropertiesString();
long zoneId = service.addZone( parentZoneId, "example.abc.com",
    propertyString );
```

Update in Java

```
APIEntity zone = service.getEntityById( zoneId );
EntityProperties properties = new EntityProperties(zone.getProperties()); //
Populate with the existing values and then update the only properties which
need to
be modified.
properties.addProperty("TextUDF", "testTextValue");
properties.addProperty("IntegerUDF", "1005");
zone.setProperties(properties.getPropertiesString());
service.update(zone);
```

Perl API Examples

Perl API example for setting UDF values when adding.

Add in Perl

```
my $newZoneId = $service->addZone( SOAP::Data->type( 'long' )->
    name( 'parentId' )->value( $newZoneId )->attr({xmlns => ''}),
    SOAP::Data->type( 'string' )->name( 'absoluteName' )->
    value( "example.abc.com" )->attr({xmlns => ''}),
```



```
SOAP::Data->type( 'string' )->name( 'properties' )->value(
  "TextUDF=testTextValue|IntegerUDF=1005" )->attr({xmlns => ''}) )->result;
```

Getting User-defined Fields

Get UDFs through API calls.

Get User-defined Field

Returns the user-defined fields information.

Output / Response

Returns the user-defined fields information.

API call:

```
APIUserDefinedField[] getUserDefinedFields ( String type, boolean requiredFieldsOnly )
```

Parameter	Description
type	The type of the user-defined fields. This must be one of the constants listed in Object Types on page 209.
requiredFieldsOnly	Specifies whether all user-defined fields of the object type will be returned or not. If set to true, only required fields will be returned.

Update Bulk User-defined Field

Updates values of various UDFs for different objects.

Output / Response

Returns a CSV file containing the following information:

LineNumber

The respective line number in the input CSV file. This appears in the first column in the output CSV file.

FailureMessage

The reason for the failure identified by the system. This appears in the second column in the output CSV file.



Note: An empty CSV file will be returned when all the rows in the input CSV file were processed successfully.

API call:

```
byte[] updateBulkUdf ( byte[] data, String properties )
```

Parameter	Description
data	<p>The file to be used to update UDFs. The file is passed to Address Manager as a byte array that is the stream of the CSV file contents. The file must follow the following pattern:</p> <ul style="list-style-type: none"> • EntityId - The object ID of the entity on which the UDF needs to be updated. This must be entered into the first column. • UDFName - The actual name of the UDF that needs to be updated. This must be entered into the second column. • newUDFValue - The new value of the UDF which needs to be updated on the entity. This must be entered into the third column. <p> Note:</p>

Parameter	Description
	<ul style="list-style-type: none"> The file format should be CSV. The file should not contain any header. The data should start from the first line. To include any special characters, users need to escape the data.
properties	Reserved for future use.

IPAM

The IP core contains information about network structures or allocation blocks and static and dynamic allocations.

This information is integrated with the DNS core to keep the DNS space current with the IP networks that it represents. DHCP configuration is modeled on the allocation blocks in the Address Manager IP core and is kept current by real-time feedback from managed servers. Dynamic DNS changes, such as address allocations, from Active Directory and other updating systems are sent to Address Manager in real time, showing administrators that an automated process made a configuration change.

IPv4 Blocks

An IPv4 block is a group of IPv4 addresses that is separated from a larger network by subnetting.

Addresses within a block cannot be routed until they have been allocated into a network. Blocks can be added and returned by IP range or CIDR notation. You can specify default DNS domains for IPv4 blocks with the **defaultDomains** property. To add a single default domain, specify the object ID for the required domain. To add multiple default domains, specify the object IDs for multiple domains as a comma-delimited list of domain object IDs.

Add IPv4 Block by CIDR

Adds a new IPv4 Block using CIDR notation.

Output / Response

Returns the object ID for the new IPv4 block.

API call:

```
long addIPv4BlockByCIDR ( long parentId, String CIDR, String properties )
```

Parameter	Description
parentId	The object ID of the target object's parent object.
CIDR	The CIDR notation defining the block (for example, 10.10/16).
properties	A string containing options. For more information about the available options, refer to IPv4Objects on page 248 in the Property Options Reference section.

Add IPv4 Block by Range

Adds a new IPv4 block defined by an address range.

Output / Response

Returns the object ID for the new IPv4 block.

API call:

```
long addIP4BlockByRange ( long parentId, String start, String end, String properties )
```

Parameter	Description
parentId	The object ID of the target object's parent object.
start	An IP address defining the lowest address or start of the block.
end	An IP address defining the highest address or end of the block.
properties	A string containing options. For more information about the available options, refer to IPv4Objects on page 248 in the Property Options Reference section.

Add Parent Block

Creates an IPv4 or IPv6 block from a list of IPv4 or IPv6 blocks or networks. All blocks and networks must have the same parent but it does not need to be contiguous.

Output / Response

Returns the object ID for the new IPv4 or IPv6 parent block. This method does not create a name for the new block.

API call:

```
void addParentBlock ( long[] blockOrNetworkIDs )
```

Parameter	Description
blockOrNetworkIDs	An array containing the object IDs of IPv4 or IPv6 blocks or networks.

Add Parent Block with Properties

Creates an IPv4 or IPv6 block with a name from a list of IPv4 or IPv6 blocks or networks. All blocks and networks must have the same parent but it does not need to be contiguous.

Output / Response

Returns the object ID for the new IPv4 or IPv6 parent block.

API call:

```
long addParentBlockWithProperties ( long[] blockOrNetworkIDs, String properties )
```

Parameter	Description
blockOrNetworkIDs	An array containing the object IDs of IPv4 or IPv6 blocks or networks.
properties	A string containing the following option: <ul style="list-style-type: none"> name-the name of the new IPv4 or IPv6 block to be created.

Get IP Range by IP Address

Returns the IPv4 Block containing the specified IPv4 address.

Use this method to find the Configuration, IPv4 Block, IPv4 Network, or DHCP Range containing a specified address. You can specify the type of object to be returned, or you can leave the type of object empty to find the most direct container for the object.

Output / Response

Returns an *APIEntity* for the object containing the specified address. If no object is found, returns an *empty APIEntity*. If *ObjectTypes.IP4Block*, *ObjectTypes.IP4Network*, or

ObjectTypes.DHCP4Range is specified as the type parameter, returns an object of the specified type. If an *empty string* ("") is specified as the type parameter, returns the most direct container for the IPv4 address.

API call:

`APIEntity getIPRangedByIP (long containerId, String type, String address)`

Parameter	Description
containerId	The object ID of the container in which the IPv4 address is located. This can be a Configuration, IPv4 Block, IPv4 Network, or DHCP Range. When you do not know the block, network, or range in which the address is located, specify the configuration.
type	The type of object containing the IPv4 or IPv6 address. Specify ObjectTypes.IP4Block , ObjectTypes.IP4Network , or ObjectTypes.DHCP4Range to find the block, network, or range containing the IPv4 address. Specify an empty string ("") to return the most direct container for the IPv4 address.
address	An IPv4 address.

Get IPv4 Block by CIDR

Returns an IPv4 Block object by calling the block using CIDR notation.

Output / Response

Returns the specified IPv4 block object from the database.

API call:

`APIEntity getEntityByCIDR (long parentId, String cidr, String type)`

Parameter	Description
parentId	The object ID of the target object's parent object.
CIDR	CIDR notation defining the block to be returned (for example, 10.10/16).
type	The type of object returned: IP4Block. This must be one of the constants listed in <i>Object Types</i> on page 209.

Get IPv4 Block by Range

Returns an IPv4 Block by calling the block using its address range.

Output / Response

Returns the requested IPv4 block object from the database.

API call:

`APIEntity getEntityByRange (long parentId, String address1, String address2, String type)`

Parameter	Description
parentId	The object ID of the target object's parent object.
address1	An IP address defining the lowest address or start of the block.
address2	An IP address defining the highest address or end of the block.

Parameter	Description
type	The type of object returned: IP4Block. This must be one of the constants listed in Object Types on page 222 .

Merge Blocks with Parent

Merges specified IPv4 blocks into a single block. The blocks must all have the same parent and must be contiguous. Blocks whose parent object is the configuration cannot contain networks.

Output / Response

None.

API call:

```
void mergeBlocksWithParent ( long[] blockIDs )
```

Parameter	Description
blockIDs	An array containing a list of IPv4 block IDs.

Merge Selected Blocks or Networks

Merges specified IPv4 blocks or IPv4 networks into a single IPv4 block or IPv4 network. The list of objects to be merged must all be of the same type (for example, all blocks or all networks). The objects must all have the same parent and must be contiguous.

Output / Response

None.

API call:

```
void mergeSelectedBlocksOrNetworks ( long[] blockOrNetworkIds, long  
blockOrNetworkToKeep )
```

Parameter	Description
blockOrNetworkIds	An array containing a list of IPv4 block or network IDs.
blockOrNetworkToKeep	The ID of the IPv4 block or IPv4 network that will retain its identity after the merge.

Update IPv4 Block

An IPv4 block's **name** property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

IPv4 Block Generic Methods

IPv4 blocks can be deleted using the generic `delete()` method.

For more information, see [Deleting Objects](#) on page 52.

IPv4 Networks

An IPv4 network is an object that attaches to a router interface that routes directly to individual IP addresses.

An IPv4 network is therefore a group of IPv4 addresses that can be routed. An IPv4 network always has a network or a block as its parent object in Address Manager. Networks can be added and returned both by IP range and CIDR notation. The next available network can also be returned and allocated. You can

specify default DNS domains for IPv4 networks with the **defaultDomains** property. To add a single default domain, specify the object ID for the required domain. To add multiple default domains, specify the object IDs for multiple domains as a comma-delimited list of domain object IDs.

Add IPv4 Network

Adds an IPv4 network using CIDR notation.

Output / Response

Returns the object ID for the new IPv4 network.

API call:

```
long addIP4Network ( long blockId, String CIDR, String properties )
```

Parameter	Description
blockId	The object ID of the new network's parent IPv4 block.
CIDR	The CIDR notation defining the network (for example, 10.10.10/24).
properties	A string containing options. For more information about the available options, please refer to IPv4Objects on page 264 in the Property Options Reference section .

Get IPv4 Range by IP Address

Returns the IPv4 Network containing the specified IPv4 address.

Use this method to find the Configuration, IPv4 Block, IPv4 Network, or DHCP Range containing a specified address. You can specify the type of object to be returned, or you can leave the type of object empty to find the most direct container for the object.

Output / Response

Returns an APIEntity for the object containing the specified address. If no object is found, returns an **empty APIEntity**. If **ObjectTypes.IP4Block**, **ObjectTypes.IP4Network**, or **ObjectTypes.DHCP4Range** is specified as the type parameter, returns an object of the specified type. If an **empty string** is specified as the **type** parameter, returns the most direct container for the IPv4 address.

API call:

```
APIEntity getIPRangedByIP ( long containerId, String type, String address )
```

Parameter	Description
containerId	The object ID of the container in which the IPv4 address is located. This can be a Configuration, IPv4 Block, IPv4 Network, or DHCP Range. When you do not know the block, network, or range in which the address is located, specify the configuration.
type	The type of object containing the IPv4 address. Specify ObjectTypes.IP4Block , ObjectTypes.IP4Network , or ObjectTypes.DHCP4Range to find the block, network, or range containing the IPv4 address. Specify an empty string to return the most direct container for the IPv4 address.
address	An IPv4 address.

Get IPv4 Network by CIDR

Returns an IPv4 Network object from the database by calling it using CIDR notation.

Output / Response

Returns the specified IPv4 network object from the database.

API call:

```
APIEntity getEntityByCIDR ( long parentId, String cidr, String type )
```

Parameter	Description
parentId	The object ID of the network's parent object.
CIDR	CIDR notation defining the network (for example, 10.10.10/24).
type	The type of object returned: IP4Network. This must be one of the constants listed in Object Types on page 209.

Get IPv4 Network by Hint

Returns an array of IPv4 networks found under a given container object. The networks can be filtered by using **ObjectProperties.hint**, **ObjectProperties.accessRight**, and **ObjectProperties.overrideType** options.


Output / Response

Returns an array of IPv4 networks based on the input argument without their properties fields populated, or returns an empty array if *containerId* is invalid. If no access right option is specified, the View access level will be used by default.

API call:

```
APIEntity[] getIP4NetworksByHint ( long containerId, int start, int count, String options )
```

Parameter	Description
containerId	The object ID for the container object. It can be the object ID of any object in the parent object hierarchy. The highest parent object is the configuration level
start	Indicates where in the list of objects to start returning objects. The list begins at an index of 0.
count	Indicates the maximum number of child objects that this method will return.
options	<p>A string containing options. The Option names available in the ObjectProperties are ObjectProperties.hint, ObjectProperties.accessRight, and ObjectProperties.overrideType. Multiple options can be separated by a (pipe) character. For example:</p> <pre>hint=ab overrideType=HostRecord accessRight=ADD</pre> <p>The values for the ObjectProperties.hint option can be the prefix of the IP address for a network or the name of a network.</p> <ul style="list-style-type: none"> Example 1 <p>The following example will match networks that have the network ID starting with 192.168. For example, 192.168.0.0/24 or 192.168.1.0/24.</p> <pre>String options = ObjectProperties.hint + "=192.168 "</pre> Example 2

Parameter	Description
	<p>The following example will match networks that have a name starting with "abc". For example, "abc", "abc123" or "abcdef".</p> <pre>String options = ObjectProperties.hint + "=abc "</pre> <p> Note: Matching networks to a network ID (Example 1) will take precedence over matching networks to a name (Example 2).</p> <p>The values for the ObjectProperties.accessRight and ObjectProperties.overrideType options must be one of the constants listed in Access Right Values on page 189 and Object Types on page 209. For example:</p> <pre>String options = ObjectProperties.accessRight + "=" + AccessRightValues.AddAccess + " " + ObjectProperties.overrideType + "=" + ObjectTypes.HostRecord;</pre>

Get IPv4 Network by Range

Returns an IPv4 Network object from the database by calling it using its address range.

Output / Response

Returns the specified IPv4 network object from the database.

API call:

```
APIEntity getEntityByRange ( long parentId, String address1, String address2, String type )
```

Parameter	Description
parentId	The object ID of the network's parent object.
address1	An IP address defining the lowest address or start of the network.
address2	An IP address defining the highest address or end of the network.
type	The type of object returned: IP4Network. This must be one of the constants listed in Object Types on page 209.

Get Next Available Network

returns the object ID for the next available (unused) network within a configuration or block.

Output / Response

Returns the object ID for the existing next available IPv4 network or, if the next available network did not exist and **autoCreate** was set to true, the newly created IPv4 network.

API call:

```
long getNextAvailableIP4Network ( long parentId, long size, boolean isLargerAllowed,
boolean autoCreate )
```

Parameter	Description
parentId	The object ID of the network's parent object.
size	The size of the network, expressed as a power of 2.

Parameter	Description
isLargerAllowed	This Boolean value indicates whether to return larger networks than those specified with the size parameter.
autoCreate	This Boolean value indicates whether the next available network should be created if it does not exist.

Get Next Available IP Range

Returns the object ID for the next available (unused) block or network within a configuration or block.

Output / Response

Returns the object ID for the existing next available IPv4 range or, if the next available IP range does not exist and **autoCreate** was set to true, the newly created IPv4 range.

API call:

`APIEntity getNextAvailableIPRange (long parentId, long size, String type, String properties)`

Parameter	Description
parentId	The object ID of the parent object under which the next available range resides (Configuration or Block).
size	The size of the range, expressed as a power of 2.
type	The type of the range object to be fetched. Currently IPv4 block and network are supported.
properties	<p>The string containing the following properties and values:</p> <ul style="list-style-type: none"> • reuseExisting—<i>True</i> or <i>False</i>. This Boolean value indicates whether to search existing empty networks to find the available IP range of specified size. • isLargerAllowed—<i>True</i> or <i>False</i>. This Boolean value indicates whether to return larger networks than those specified with the size parameter. • autoCreate—<i>True</i> or <i>False</i>. This Boolean value indicates whether the next available IP range should be created in the parent object if it does not exist. • traversalMethod—This parameter identifies the appropriate search algorithm to find the suitable object. The possible values are: <ul style="list-style-type: none"> • TraversalMethodology.NO_TRAVERSAL (NO_TRAVERSAL)—will attempt to find the next range directly under the specified parent object. It will not search through to the lower level objects. • TraversalMethodology.DEPTH_FIRST (DEPTH_FIRST)—will attempt to find the next range under the specified object by iteratively through its children one by one. After exploring the object recursively for its child ranges, it will move to the next child object. • TraversalMethodology.BREADTH_FIRST (BREADTH_FIRST)—will attempt to find the next range under the specified object by iterative levels. It will first find the range immediately below the specified parent object. If not found, then it will attempt to find the range under all the first child objects.

Get Next Available IP Ranges

Returns the object IDs for the next available (unused) blocks or networks within a configuration or block.

Output / Response

Returns consecutive matching IPv4 range object IDs. If the next available ranges do not exist and you have set the **autoCreate** property to *true*, new IPv4 ranges will be created and their object IDs will be returned.

API call:

```
APIEntity[] getNextAvailableIPRanges ( long parentId, long size, String type, int count, String properties )
```

Parameter	Description
parentId	The object ID of the parent object under which the next available range resides (Configuration or Block).
size	The size of the range, expressed as a power of 2.
type	The type of the range object to be fetched. Currently only IPv4 network is supported.
count	<p>The number of networks to be found.</p> <p>➔ Note: If the number of networks count is greater than 1:</p> <ul style="list-style-type: none"> isLargerAllowed and traversalMethod properties will not be applicable. The DEPTH_FIRST methodology will be used to search objects.
properties	<p>The string containing the following properties and values:</p> <ul style="list-style-type: none"> reuseExisting—<i>True</i> or <i>False</i>. This Boolean value indicates whether to search existing empty networks to find the available IP range of specified size. isLargerAllowed—<i>True</i> or <i>False</i>. This Boolean value indicates whether to return larger networks than those specified with the sizeparameter. autoCreate—<i>True</i> or <i>False</i>. This Boolean value indicates whether the next available IP range should be created in the parent object if it does not exist. traversalMethod—This parameter identifies the appropriate search algorithm to find the suitable object. The possible values are: <ul style="list-style-type: none"> TraversalMethodology.NO_TRAVERSAL (NO_TRAVERSAL)—will attempt to find the next range directly under the specified parent object. It will not search through to the lower level objects. TraversalMethodology.DEPTH_FIRST (DEPTH_FIRST)—will attempt to find the next range under the specified object by iteratively through its children one by one. After exploring the object recursively for its child ranges, it will move to the next child object. TraversalMethodology.BREADTH_FIRST (BREADTH_FIRST)—will attempt to find the next range under the specified object by iterative levels. It will first find the range immediately below the specified parent object. If not found, then it will attempt to find the range under all the first child objects.

Split IPv4 Network

Splits an IPv4 network into the specified number of networks.

Output / Response

Returns an array of networks created after splitting the network.

API call:

```
APIEntity[] splitIP4Network ( long networkId, int numberOfParts, String options )
```

Parameter	Description
networkId	The database object ID of the network that is being split.
numberOfParts	The number of the networks into which the network will be split. Valid values are 2 to the power of 2 up to 1024.
options	<p>A string containing the following options:</p> <ul style="list-style-type: none"> • assignDefaultGateway—a Boolean value. If set to <i>true</i>, a gateway will be created by using the default gateway value which is the first IP address in the network. If set to <i>false</i>, no gateway will be created. The default value is <i>true</i>. • overwriteConflicts—a Boolean value. If set to <i>true</i>, any conflicts within the split IPv4 network will be removed. The default value is <i>false</i>. • template—a network template ID. The default value is 0 which means no network template will be used. Specify a network template ID if you wish to apply one. • preserveGateway—a Boolean value. If set to <i>true</i>, the gateway in the original network will be preserved. The default value is <i>true</i>.

Update IPv4 Network

An IPv4 network's **name** property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

IPv4 Network Generic Methods

IPv4 networks can be deleted using the generic `delete()` method.

For more information, see [Deleting Objects](#) on page 52.

IPv4 Network Templates

IPv4 network templates allow you to create standard settings that can be applied when you create new networks.

Whenever you change template settings, all networks based on the template are updated accordingly. Use network templates to standardize address assignments and DHCP options.

Address Manager API v4.0 and greater includes the following changes to IPv4 network templates when performing Add, Update and Get operations for IPv4 Network Templates through API:

- The pipe (|) separator in the sub-type value has been replaced with commas (,).
- If the name properties contains commas (,) or back-slashes (\), it needs to be escaped with backward slash (\).

Add IPv4 Network Template

Add an IPv4 network template to the specified configuration.

Output / Response

Returns the object ID of the new IPv4 network template.

API call:

```
long addIP4NetworkTemplate ( long configurationId, String name, String properties )
```

Parameter	Description
configurationId	The object ID of the configuration in which the IPv4 template is located.
name	The name of the IPv4 network template. This value cannot be empty or null.
properties	A string defining the IPv4 network template properties. For example, <code>gateway=[gateway_offset] reservedAddresses={type,offset,size,direction,name}</code> . Refer to the Properties lists table.

Properties lists

Property	Value
gateway_offset	This is to specify which address to assign an IPv4 gateway. When there is a negative sign in front of the gateway offset, then the gateway is at the end of the range. For example, if the value of gateway offset is $-n$, the n^{th} IP address from the end of range will be the gateway.
type	Can be either RESERVED_BLOCK or RESERVED_DHCP_RANGE .
offset	This is to specify from which address to start to assign IPv4 addresses.
size	The size of the network.
direction	Can be either FROM_START or FROM_END .
name	The name of the network.

Assign or Update Template

Assigns, updates, or removes DNS zone and IPv4 network templates.


Output / Response

None.

API call:

```
void assignOrUpdateTemplate ( long entityId, long templateId, String properties )
```

Parameter	Description
entityId	The object ID of the IPv4 network to which the network template is to be assigned or updated, or the object ID of the zone to which the zone template is to be assigned or updated.
templateId	The object ID of the DNS zone template or IPv4 network template. To remove a template, set this value to 0 (zero).
properties	A string containing the following settings: <ul style="list-style-type: none"> <i>ObjectProperties.templateType</i> - Specifies the type of template on which this operation is being performed. This is mandatory. The possible values are <i>ObjectProperties.IP4NetworkTemplateType</i> (Assigning or updating <i>IP4NetworkTemplate</i> on an <i>IP4Network</i>) and <i>ObjectProperties.zoneTemplateType</i> (Assigning or updating <i>zoneTemplate</i> on a <i>DNS zone</i>). <p>Along with <i>ObjectProperties.templateType</i>, you can also specify the reapply mode for various properties of the template.</p>

Parameter	Description
	<p>For Network template, the following additional parameters can also be specified:</p> <ul style="list-style-type: none"> ObjectProperties.gatewayReapplyMode ObjectProperties.reservedAddressesReapplyMode ObjectProperties.dhcpRangesReapplyMode ObjectProperties.ipGroupsReapplyMode ObjectProperties.optionsReapplyMode <p>For Zone Template, the following additional parameter can also be specified:</p> <ul style="list-style-type: none"> ObjectProperties.zoneTemplateReapplyMode <p>The possible values for re-apply mode properties are:</p> <ul style="list-style-type: none"> ObjectProperties.templateReapplyModeUpdate ObjectProperties.templateReapplyModeIgnore ObjectProperties.templateReapplyModeOverwrite <p>If the re-apply mode is not specified in the properties, the default ObjectProperties.templateReapplyModeIgnore mode is used.</p> <p> Note: If you are not using Java or Perl, refer to Object Properties on page 201 for the actual values.</p>

Java client example

```
EntityProperties ntProp = new EntityProperties();
ntProp.addProperty( ObjectProperties.templateType,
ObjectProperties.IP4NetworkTemplateType );
ntProp.addProperty( ObjectProperties.gatewayReapplyMode,
ObjectProperties.templateReapplyModeUpdate );
ntProp.addProperty( ObjectProperties.reservedAddressesReapplyMode,
ObjectProperties.templateReapplyModeUpdate );
service.assignOrUpdateTemplate( ip4N20_26Id, networkTemplateId,
ntProp.getPropertiesString() );
```

Perl client example

```
SOAP::Data->type( 'string' )->name( 'properties' )->
value( ObjectProperties::templateType."=".ObjectProperties::
IP4NetworkTemplateType."|".
ObjectProperties:: gatewayReapplyMode."=".ObjectProperties::
templateReapplyModeUpdate."|" )
->attr({xmlns => ''})->result;
```

Re-apply Template


Reapplies IPv4 network templates. The template must already be applied to an object before you can re-apply or remove it.

Output / Response

None.

API call:

```
void reapplyTemplate ( long templateId,String properties )
```

Parameter	Description
templateId	The object ID of the IPv4 network template or DNS zone template to be assigned or updated.
properties	<p>A string containing the following settings:</p> <ul style="list-style-type: none"> The properties value must include ObjectProperties.templateType with the value of ObjectProperties.IP4NetworkTemplateType or ObjectProperties.zoneTemplateType. To re-apply the network gateway in a IPv4 network template, include ObjectProperties.gatewayReapplyMode. This is optional. If the re-apply mode is not specified in the properties, the default ObjectProperties.templateReapplyModeIgnore mode is used. The available re-apply modes include: <ul style="list-style-type: none"> ObjectProperties.templateReapplyModeUpdate ObjectProperties.templateReapplyModeIgnore ObjectProperties.templateReapplyModeOverwrite <p> Note:</p> <ul style="list-style-type: none"> ObjectProperties.templateReapplyModeOverwrite is not applicable for <i>Gateway</i> and <i>Reserved Addresses</i>. Use ObjectProperties.templateReapplyModeUpdate instead to update. ObjectProperties.templateReapplyModeUpdate is not applicable for <i>Reserved DHCP Ranges</i>, <i>IP Groups</i> and <i>Zone Templates</i>. Use ObjectProperties.templateReapplyModeOverwrite instead to update. Both ObjectProperties.templateReapplyModeUpdate and ObjectProperties.templateReapplyModeOverwrite are applicable for <i>Deployment Options</i>.

Java client example

```
EntityProperties ntProp = new EntityProperties();
ntProp.addProperty( ObjectProperties.templateType,
ObjectProperties.IP4NetworkTemplateType );
ntProp.addProperty( ObjectProperties.gatewayReapplyMode,
ObjectProperties.templateReapplyModeUpdate );
ntProp.addProperty( ObjectProperties.reservedAddressesReapplyMode,
ObjectProperties.templateReapplyModeUpdate );
service.reapplyTemplate( networkTemplateId3, ntProp.getPropertiesString() );
```

Perl client example

```
SOAP::Data->type( 'string' )->name( 'properties' )->
value( ObjectProperties::templateType."=".ObjectProperties::
IP4NetworkTemplateType."|".
ObjectProperties:: gatewayReapplyMode."=".ObjectProperties::
templateReapplyModeUpdate."|" )
->attr({xmlns => ''}) ->result;
```

Update IPv4 Network Template Name

An IPv4 network template's name property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

IPv4 Network Template Generic Methods

IPv4 network templates use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

IPv4 addresses

An address is the actual IP address leased or assigned to a member of a network.

IPv4 addresses need to be assigned a particular allocation rather than simply be added. The address allocation can be checked, along with any host records that are dependent on it. The next available address can also be returned. Only addresses within an existing network can be assigned.

Assign IPv4 Address

Assigns a MAC address and other properties to an IPv4 address.


Output / Response

Returns the object ID for the newly assigned IPv4 address.

API call:

```
long assignIP4Address ( long configurationId, String ip4Address, String macAddress, String hostInfo,
String action, String properties)
```

Parameter	Description
configurationId	The object ID of the configuration in which the IPv4 address is located.
ip4Address	The IPv4 address.
macAddress	The MAC address to assign to the IPv4 address. The MAC address can be specified in the format <code>nnnnnnnnnnnnnn</code> , <code>nn-nn-nn-nn-nn-nn</code> or <code>nn:nn:nn:nn:nn:nn</code> , where <code>nn</code> is a hexadecimal value.
hostInfo	<p>A string containing host information for the address in the following format:</p> <pre>hostname,viewId,reverseFlag,sameAsZoneFlag[, hostname,viewId,reverseFlag,sameAsZoneFlag,...]</pre> <p>Where:</p> <ul style="list-style-type: none"> hostname - The Fully Qualified Domain Name (FQDN) for the host record to be added. viewId - The object ID of the view under which this host should be created. reverseFlag - The flag indicating if a reverse record should be created. The possible values are true or false. sameAsZoneFlag - The flag indicating if record should be created as same as zone record. The possible values are true or false. <p>The comma-separated parameters may be repeated in the order shown above. The string must not end with a comma.</p>
action	This parameter must be set to one of the constants shown in IP Assignment Action Values on page 201.
properties	A string containing the following property, including user-defined fields:

Parameter	Description
	<ul style="list-style-type: none"> ptrs—a string containing the list of unmanaged external host records to be associated with the IPv4 address in the following format: <pre>viewId,exHostFQDN[, viewId,exHostFQDN,...]</pre> <p>You can assign External Host records to an IPv4 address using the following method:</p> <pre>EntityProperties props = new EntityProperties(); props.addProperty(ObjectProperties.ptrs, "123,exHostFQDN.com,456,exHostFQDN.net") long addressId = service.assignIP4Address(configurationId, IPv4Address, macAddressStr, hostInfo, IPAssignmentActionValues.MAKE_STATIC, props.getPropertiesString());</pre> name—name of the IPv4 address. locationCode—the hierarchical location code consists of a set of 1 to 3 alpha-numeric strings separated by a space. The first two characters indicate a country, followed by next three characters which indicate a city in UN/LOCODE. New custom locations created under a UN/LOCODE city are appended to the end of the hierarchy. For example, CA TOR OF1 indicates: CA= Canada TOR=Toronto OF1=Office 1. <p> Note: The code is case-sensitive. It must be all UPPER CASE letters. The country code and child location code should be alphanumeric strings.</p>

Assign Next Available IPv4 Address

Assigns a MAC address and other properties to the next available and unallocated IPv4 address within a configuration, block, or network.


Output / Response

Returns the object ID for the newly assigned IPv4 address.

API call:

`APIEntity assignNextAvailableIP4Address (long configurationId, long parentId, String macAddress, String hostInfo, String action, String properties)`

Parameter	Description
configurationId	The object ID of the configuration in which the IPv4 address is located.
parentId	The object ID of the configuration, block, or network in which to look for the next available address.
macAddress	The MAC address to assign to the IPv4 address. The MAC address can be specified in the format <code>nnnnnnnnnnnnnn</code> , <code>nn-nn-nn-nn-nn-nn</code> or <code>nn:nn:nn:nn:nn:nn</code> , where <code>nn</code> is a hexadecimal value.
hostInfo	<p>A string containing host information for the address in the following format:</p> <pre>hostname,viewId,reverseFlag,sameAsZoneFlag[, hostname,viewId,reverseFlag,sameAsZoneFlag,...]</pre> <p>Where:</p>

Parameter	Description
	<ul style="list-style-type: none"> • hostname - The Fully Qualified Domain Name (FQDN) for the host record to be added. • viewId - The object ID of the view under which this host should be created. • reverseFlag - The flag indicating if a reverse record should be created. The possible values are true or false. • sameAsZoneFlag - The flag indicating if record should be created as same as zone record. The possible values are true or false. <p>The comma-separated parameters may be repeated in the order shown above. The string must not end with a comma.</p>
action	This parameter must be set to one of the constants shown in IP Assignment Action Values on page 201.
properties	<p>A string containing the following property, including user-defined fields:</p> <ul style="list-style-type: none"> • ptrs—a string containing the list of unmanaged external host records to be associated with the IPv4 address in the following format: <div data-bbox="548 751 1463 810" data-label="Text"> <pre>viewId,exHostFQDN[, viewId,exHostFQDN,...]</pre> </div> <p>You can assign External Host records to an IPv4 address using the following method:</p> <div data-bbox="548 909 1463 1163" data-label="Text"> <pre>EntityProperties props = new EntityProperties(); props.addProperty(ObjectProperties.ptrs, "123,exHostFQDN.com,456,exHostFQDN.net") long addressId = service.assignIP4Address(configurationId, IPv4Address, macAddressStr, hostInfo, IPAssignmentActionValues.MAKE_STATIC, props.getPropertiesString());</pre> </div> • name—name of the IPv4 address. • locationCode—the hierarchical location code consists of a set of 1 to 3 alpha-numeric strings separated by a space. The first two characters indicate a country, followed by next three characters which indicate a city in UN/LOCODE. New custom locations created under a UN/LOCODE city are appended to the end of the hierarchy. For example, CA TOR OF1 indicates CA= Canada TOR=Toronto OF1=Office 1. <p> Note: The code is case-sensitive. It must be all UPPER CASE letters. The county code and child location code should be alphanumeric strings.</p> <ul style="list-style-type: none"> • skip—This is <i>optional</i>. Use to specify the IP address ranges or IP addresses to skip. You can specify multiple IP addresses separated by comma. To specify the range between the start and end addresses, use a hyphen (-). For example, <code>skip=192.0.2.128-192.0.2.222,192.0.2.223</code>. • offset—This is <i>optional</i>. Use to specify from which address to start to assign IPv4 address. For example, <code>offset=192.0.2.100</code>. • excludeDHCPRange—<i>true</i> or <i>false</i>. To specify whether IP addresses in DHCP ranges should be excluded from assignment. The default value is <i>false</i>. For example, <code>offset=192.0.2.100 excludeDHCPRange=true</code>.

Get IPv4 Address

Returns the details for the requested IPv4 address object.

Output / Response

Returns the requested IPv4 Address object from the database.

API call:

```
APIEntity getIP4Address ( long containerId, String address)
```

Parameter	Description
containerId	The object ID for the configuration, block, network, or DHCP range in which this address is located.
address	The IPv4 address.

Get Next IPv4 Address

Returns the next available IP addresses in octet notation under specified circumstances.

Output / Response

Returns the IPv4 address in octet notation.

API call:

```
String getNextIP4Address ( long parentId, String properties)
```

Parameter	Description
parentId	The network or configuration Id.
properties	<p>The property string contains three properties; skip, offset and excludeDHCPRange. The values for skip and offset must be IPv4 addresses and must appear in dotted octet notation.</p> <ul style="list-style-type: none"> • skip - This is optional. It is used to specify the IP address ranges or IP addresses to skip, separated by comma. A hyphen(-), not a dash is used to separate the start and end addresses. <ul style="list-style-type: none"> ➔ Note: Do not use the skip property if the parentId is a configuration Id. If you do, an error message appears, 'Skip is not allowed for configuration level'. • offset - This is optional. This is to specify from which address to start to assign IPv4 Address. • excludeDHCPRange - This specifies whether IP addresses in DHCP ranges should be excluded from assignment. The value is either <i>true</i> or <i>false</i>, default value is <i>false</i>. <pre>skip=10.10.10.128-10.10.11.200,10.10.11.210 offset=10.10.10.100 excludeDHCPRange=true </pre>

Check Allocation for IPv4 Address

Returns the allocation information for an IPv4 DHCP allocated address.

Output / Response

Returns a Boolean value indicating whether the address is allocated.

API call:

```
boolean isAddressAllocated ( long configurationId, String ipAddress, String macAddress )
```

Parameter	Description
configurationId	The object ID for the configuration in which the IPv4 DHCP allocated address is located.
ipAddress	The IPv4 DHCP allocated address.
macAddress	The MAC address associated with the IPv4 DHCP allocated address. The MAC address can be specified in the format <i>nnnnnnnnnnnnnn</i> , <i>nn-nn-nn-nn-nn-nn</i> or <i>nn:nn:nn:nn:nn:nn</i> , where <i>nn</i> is a hexadecimal value.

Get Next Available Address

Returns the IPv4 address for the next available (unallocated) address within a configuration, block, or network.

Output / Response

Returns the next available IPv4 address in an existing network as a string.

API call:

```
String getNextAvailableIP4Address ( long parentId )
```

Parameter	Description
parentId	The object ID for configuration, block, or network in which to look for the next available address.

Update IPv4 Address

An IPv4 address's name property can be updated using the generic `update()` method.

For more information, refer to [Updating Objects](#) on page 50.

IPv4 Address Generic Methods

IPv4 addresses can be deleted using the generic `delete()` method.

For more information, see [Deleting Objects](#) on page 52.

Change IPv4 Address State

Converts the state of an address from and between Reserved, DHCP Reserved, and Static, or DHCP Allocated to DHCP Reserved.

Output / Response

Converts an IP address from its current state to a target state; statically assigned, DHCP reserved, or logically reserved (non-DHCP). For example, this method can convert an IP address from a logical reservation to a static assignment or vice versa.

API call:

```
void changeStateIP4Address ( long addressId, String targetState, String macAddress )
```

Parameter	Description
addressId	The database ID of the address object.
targetState	One of MAKE_STATIC , MAKE_RESERVED , MAKE_DHCP_RESERVED . All of these constants are defined in the Java Class <code>IPAssignmentActionValues</code> or in the <i>API.pm</i> file for Perl.

Parameter	Description
macAddress	Optional and only needed, if the target requires it. (e.g. MAKE_DHCP_RESERVED)

Additional IP Addresses

Add multiple IPv4 addresses to the Services interface or loopback address for DNS services.

Multiple DNS service addresses provide flexibility and centralized control when consolidating old DNS servers into one single server without disrupting any configurations that might be using the old IP addresses.



Note:

- You can configure a maximum of 400 IP addresses per DNS/DHCP Server appliance or VM, including IPv4, IPv6, IPv6 link-local addresses, and loopback addresses.
- IP addresses that you are adding must be unique and must not conflict with other IP addresses used by the server.



Attention: VLAN interfaces are not currently supported by the following API methods:

- `void addAdditionalIPAddresses()`
- `void removeAdditionalIPAddresses()`
- `String getAdditionalIPAddresses()`

Add Additional IP Addresses

Adds additional IPv4 addresses and loopback addresses to the Services interface for DNS service.

Output / Response

None.



Attention: This method does not support VLAN interfaces.

API call:

```
void addAdditionalIPAddresses ( long serverId, String ipsToAdd, String properties )
```

Parameter	Description
serverId	The database object ID of the server to which additional services IP address will be added.
ipsToAdd	The list of IP addresses to be added. You can specify multiple IP addresses with a separator (). The supported format is <code>[IP, serviceType IP, serviceType]</code> . For example, <code>10.0.0.10/32,loopback 11.0.0.3/24,service 12.0.0.3,loopback</code> .
properties	Adds object properties. Currently there is no supported properties. Reserved for future use.

Remove Additional IP Addresses

Removes additional IPv4 addresses and loopback addresses from the Services interface.

Output / Response

None.



Attention: This method does not support VLAN interfaces.

API call:

```
void removeAdditionalIPAddresses ( long serverId, String ipsToRemove, String properties )
```

Parameter	Description
serverId	The database object ID of the server from which additional services IP addresses need to be removed.
ipsToRemove	The list of IP addresses to be removed. You can specify multiple IP addresses with a separator (). The supported format is [IP, serviceType IP, serviceType]. For example, 10.0.0.10/32,loopback 11.0.0.3/24,service 12.0.0.3,loopback.
properties	Adds object properties. Currently there is no supported properties. Reserved for future use.

Get Additional IP Addresses

Returns IPv4 addresses and loopback addresses added to the Service interface for DNS services.

Output / Response

Returns the list of additional IP addresses configured on the server in the following format: **[IP,serviceType|IP,serviceType]**. For example, 10.0.0.10/32,loopback|11.0.0.3/24,service|12.0.0.3/32,loopback.



Attention: This method does not support VLAN interfaces.

API call:

```
String getAdditionalIPAddresses ( long adonisID, String properties )
```

Parameter	Description
adonisID	The database object ID of the server on which additional services IP address have been added.
properties	The supported property is: <ul style="list-style-type: none"> serviceType—type of service for which a list of IP addresses will be retrieved. Available types are AdditionalIPServiceType.SERVICE and AdditionalIPServiceType.LOOPBACK. If serviceType is not provided, all additional IP addresses of the services interface will be returned.

IPv4 Group

IP grouping dedicates a set of IP addresses to a certain group in order to limit a user's accessibility to these IP addresses, depending on the user's access rights.

IP grouping helps you better manage IP addresses and troubleshoot easily when you have issues with IP addresses. You can define the range of IP addresses and grant access rights to a certain user or a group. The specified user or group can only access the IP addresses defined in the IP group and manipulate as needed. This feature is especially useful when there are multiple IP networks with large number of addresses. You can group the IP addresses based on user, department, or tasks.

Add IPv4 IP Group by Range

Adds an IPv4 IP group by range bounds; start address and end address.

Output / Response

Returns the object ID for the new IPv4 IP group range.

API call:

```
long addIP4IPGroupByRange ( long parentId, String name, String start, String end, String properties )
```

Parameter	Description
parentId	The object ID for the network in which this IP group is located.
name	The name of the IP group.
start	A start IP address of the IP group range.
end	An end IP address of the IP group range.
properties	Adds object properties, including the user-defined fields.

Add IPv4 IP Group by Size

Adds an IPv4 IP group by size.

Output / Response

returns the object ID for the new IPv4 IP group range.

API call:

```
long addIP4IPGroupBySize ( long parentId, String name, int size, String positionRangeBy, String positionValue, String properties )
```

Parameter	Description
parentId	The object ID for the network in which this IP group is located.
name	The name of the IP group.
size	The number of addresses in the IP group.
positionRangeBy	A string specifying the position of the IP group range in the parent network. The value must be one of the constants listed in PositionRangeBy on page 225. This is optional. If specified, positionValue must be provided.
positionValue	The offset value when using positionRangeBy.START_OFFSET or positionRangeBy.END_OFFSET . The start address of the IP group in the network when using positionRangeBy.START_ADDRESS . This is required only if positionRangeBy is specified.
properties	Adds object properties, including the user-defined fields.

IPv4 Objects

IPv4 objects include blocks, networks, DHCP ranges, templates, addresses and reconciliation policy.

Move IPv4 Object (deprecated)

Moves an IPv4 block, an IPv4 network, or an IPv4 address to a new IPv4 address.

Output / Response

None.



Attention: This method is deprecated in favor of the more extensive **Move IP Object** method. Use `moveIPObject()` instead.



Note: The block or network being moved must fit fully within the new parent object and must also not overlap its sibling objects. A network object cannot be moved directly beneath a configuration as it must be a child of a block object.

API call:

```
void moveIP4Object ( long objectId, String address )
```

Parameter	Description
objectId	The object ID of the IPv4 block, network, or IP address to be moved.
address	The new address for the IPv4 block, network, or IP address.

Move IP Object

Moves an IPv4 block, IPv4 network, IPv4 address, IPv6 block or IPv6 network to a new IPv4 or IPv6 address.

Output / Response

None.



Note: This method is a more extensive version of the **Move IPv4 Object** method that this method replaces. Use this method to move various supported IP objects.

API call:

```
void moveIPObject ( long objectId, String address, String options )
```

Parameter	Description
objectId	The ID of the object to be moved. Currently IPv4 blocks, IPv4 networks, IPv4 addresses, IPv6 blocks and IPv6 networks are supported.
address	The new address for the object.
options	<p>A string containing the noServerUpdate option.</p> <ul style="list-style-type: none"> noServerUpdate - A boolean value. If set to <i>true</i>, instant dynamic host record changes will not be performed on DNS/DHCP Servers when moving an IPv4 address object. <p> Note: noServerUpdate works only for an IPv4 address object.</p>

Resize Range

Changes the size of an IPv4 block, IPv4 network, DHCPv4 range, IPv6 block or IPv6 network.


Output / Response

None.

API call:

```
void resizeRange ( long objectId, String range, String options )
```

Parameter	Description
objectId	The ID of the object to be resized. Currently IPv4 block, IPv4 network, DHCPv4 range, IPv6 Block and IPv6 Network are supported.
range	<p>The new size for the object to be resized.</p> <p>For the IPv4 block, IPv4 network or DHCPv4 range, specify the size in CIDR notation or as an address range in the <i>ipAddressStart-ipAddressEnd</i> format.</p> <p>For the IPv6 block or IPv6 network, specify the size in <i>Starting address/Size</i> format.</p>

Parameter	Description
options	<p>A string containing the following options:</p> <ul style="list-style-type: none"> • ObjectProperties.convertOrphanedIPAddressesTo <p> Note: This option applies only to DHCPv4 range.</p> <p>The possible values are:</p> <ul style="list-style-type: none"> • <i>STATIC</i> • <i>DHCP_RESERVED</i> • <i>UNALLOCATED</i> <p>For example:</p> <pre>service.resizeRange(<rangeID>, <"ipAddressStart-ipAddressEnd">, "convertOrphanedIPAddressesTo=<value>");</pre> <ul style="list-style-type: none"> • If the value is an empty string (""), the default is <i>DHCP_RESERVED</i>. • If the option value is incorrect, an exception will be thrown. • If the option name is incorrect, the option will be ignored. Therefore, orphaned IP addresses will remain as assigned.

IPv4 Discovery and Reconciliation

IPv4 address discovery and reconciliation service in Address Manager discovers IPv4 addresses on your network.

Using a specifically defined discovery engine in the policy, you can explore your network from one or multiple routers and layer 3 switches for a host's IPv4 addresses, hardware addresses, interfaces, VLAN and port information, and their DNS host names if DNS is available. The discovered results of a reconciliation policy are stored in a JSON file in the Address Manager file system and then read by the reconciliation service to compare the result with the existing IPv4 addresses in Address Manager.

After running IPv4 reconciliation policies, JSON files that contain discovered results are kept in the Address Manager file system until the API method is run again. You can use the following API methods to access this discovery result in the JSON files:

- [Get Discovered Devices](#) on page 83
- [Get Discovered Device](#) on page 83
- [Get Discovered Device Interfaces](#) on page 83
- [Get Discovered Device Networks](#) on page 83
- [Get Discovered Device Hosts](#) on page 84
- [Get Discovered Device Vlans](#) on page 84
- [Get Discovered Device ARP Entries](#) on page 84
- [Get Discovered Device MAC Address Entries](#) on page 85

Add IPv4 Reconciliation Policy

Adds an IPv4 reconciliation policy.

Output / Response

Adds an IPv4 reconciliation policy.

API call:

```
long addIP4ReconciliationPolicy ( long parentId, string name, string properties )
```


Parameter	Description
parentId	The object ID of the parent network of the policy. You can create IPv4 reconciliation policies at the configuration, IPv4 block or IPv4 network levels.
name	The name of the policy.
properties	A string containing properties and values listed in List of properties and values .

List of properties and values

discoveryType	A type of discovery method to use for the network discovery operation. This must be one of the constants listed in IP Discovery Type on page 201.
seedRouterAddress	IPv4 address. This is not the Default Gateway Address.
snmpVersion	Constants defined in SNMP Version. This must be one of the constants listed in SNMP Version on page 213.
snmpPortNumber	An integer greater than 0.
snmpCommunityString	Strings separated by comma (,). For example, <i>community10,community12,community13</i> .
securityLevel	This must be one of the constants listed in SNMP Security Levels on page 214.
context	A string. This is required only when snmpVersion is v3.
authenticationType	Constants defined in SNMP Authentication Type. This must be one of the constants listed in SNMP Authentication Type on page 214. This is required only when securityLevel is AUTH_NOPRIV or AUTH_PRIV.
authPassphrase	A string. This is required only when securityLevel is AUTH_NOPRIV or AUTH_PRIV.
privacyType	A string containing the privacy encryption types. This must be one of the constants listed in SNMP Privacy Type on page 214. This is required only when securityLevel is AUTH_PRIV.
privacyPassphrase	A string containing the privacy authentication password. This is required only when securityLevel is AUTH_PRIV.
networkBoundaries	IPv4 ranges separated by comma(.). CIDR and IPv4 range formats are supported. For example, 10.0/8,13.0.0.1-13.0.0.126.
blackHoleVlan	VLAN ID for the black hole VLAN. This is used as a default VLAN for all unused ports.
trunkDefaultVlan	Unused VLAN ID to be assigned to a trunk as a native/default VLAN to protect controlled traffic from being spoofed.
skipFqdn	<i>True</i> or <i>false</i> . This is used to determine whether Address Manager discovery engine to perform FQDN resolution and DNS lookups against any DNS server.
dnsServers	One or more IPv4 or IPv6 address separated by comma (,). This is used to perform FQDN and DNS reverse lookups.
enableLayer2Discovery	<i>True</i> or <i>false</i> .

schedule	<p>Schedule settings:</p> <ul style="list-style-type: none"> • Format: hh:mm,dd MMM yyyy, frequencyType • frequency: frequencyPeriod • frequencyType: EVERY or ONCE • frequency: an integer greater than 0 • frequencyPeriod: constant defined in TimeUnits which includes [MINUTES HOURS DAYS] <p>For example, <i>03:37am,31 May 2011,EVERY,6,Days</i></p>
activeStatus	<i>True or false.</i>
acceptanceCriteriaReclaim	<p>Used to enable automated acceptance. acceptanceCriteriaReclaim, acceptanceCriteriaUnknown, acceptanceCriteriaMismatch and view should be used together as a complete configuration.</p> <ul style="list-style-type: none"> • Format: timeValue, timeUnit, actionType • timeValue: a integer greater than 0 • timeUnit: constant defined in timeUnits which includes [MINUTES HOURS DAYS] • actionType: constant defined in AcceptanceActionType which includes [RECONCILE, NOACTION] <p>For example, <i>10,MINUTES,RECONCILE</i></p>
acceptanceCriteriaUnknown	<p>Used to enable automated acceptance.</p> <ul style="list-style-type: none"> • Format: timeValue, timeUnit, actionType • timeValue: a integer greater than 0 • timeUnit: constant defined in timeUnits which includes [MINUTES HOURS DAYS] • actionType: constant defined in AcceptanceActionType which includes [RECONCILE, NOACTION] <p>For example, <i>20,HOURS,NOACTION</i></p>
acceptanceCriteriaMismatch	<p>Used to enable automated acceptance.</p> <ul style="list-style-type: none"> • Format: timeValue, timeUnit,actionType • timeValue: a integer greater than 0 • timeUnit: constant defined in timeUnits which includes [MINUTES HOURS DAYS] • actionType: constant defined in AcceptanceActionType which includes [RECONCILE, NOACTION] <p>For example, <i>30,MINUTES,RECONCILE</i></p>
view	This is used if a user wants to enable the automated acceptance for an existing view's name.
overrideList	IPv4 ranges separated by comma(.). CIDR and IP4 range formats are supported. For example, <i>10/16,172/16,172.25.0.2-172.25.0.18.</i>

IPv4 Discovery and Reconciliation Generic Methods

IPv4 Discovery and Reconciliation use the generic `get()`, `update()` and `delete()` methods for entities.

For more information, refer to [Getting Objects](#) on page 44, [Updating Objects](#) on page 50 and [Deleting Objects](#) on page 52.

Get Discovered Devices

Returns a list of discovered Layer 2 or Layer 3 devices by running an IPv4 reconciliation policy specified.

Output / Response

Returns an array of discovered Layer 2 or Layer 3 devices.

To understand the returned result, refer to the constants listed in [Device Properties](#) on page 190.

API call:

```
APIEntity[] getDiscoveredDevices ( long policyId )
```

Parameter	Description
policyId	The object ID for the IPv4 reconciliation policy.

Get Discovered Device

Returns the object ID of the discovered device by running an IPv4 reconciliation policy.

Output / Response

Returns the object ID of the discovered device.

To understand the returned result, refer to the constants listed in [Device Properties](#) on page 190.

API call:

```
APIEntity getDiscoveredDevice ( long policyId, long deviceId )
```

Parameter	Description
policyId	The object ID for the IPv4 reconciliation policy.
deviceId	The object ID of the discovered device.

Get Discovered Device Interfaces

Returns all interfaces of a specific device discovered by running an IPv4 reconciliation policy.

Output / Response

Returns all interfaces of a specific device.

To understand the returned result, refer to the constants listed in [Device Properties](#) on page 190.

API call:

```
APIEntity[] getDiscoveredDeviceInterfaces ( long policyId, long deviceId )
```

Parameter	Description
policyId	The object ID for the IPv4 reconciliation policy.
deviceId	The object ID of the discovered device.

Get Discovered Device Networks

Returns all networks of a specific device discovered by running an IPv4 reconciliation policy.

Output / Response

Returns all networks of a specific device.

To understand the returned result, refer to the constants listed in [Device Properties](#) on page 190.

API call:

```
APIEntity[] getDiscoveredDeviceNetworks ( long policyId, long deviceId )
```

Parameter	Description
policyId	The object ID for the IPv4 reconciliation policy.
deviceId	The object ID of the discovered device.

Get Discovered Device Hosts

Returns all hosts of a specific device discovered by running an IPv4 reconciliation policy.

Output / Response

Returns all hosts of a specific device.

To understand the returned result, refer to the constants listed in [Device Properties](#) on page 190.

API call:

```
APIEntity[] getDiscoveredDeviceHosts ( long policyId, long deviceId )
```

Parameter	Description
policyId	The object ID for the IPv4 reconciliation policy.
deviceId	The object ID of the discovered device.

Get Discovered Device Vlans

Returns all Vlans of a specific device discovered by running an IPv4 reconciliation policy.

Output / Response

Returns all Vlans of a specific device.

To understand the returned result, refer to the constants listed in [Device Properties](#) on page 190.

API call:

```
APIEntity[] getDiscoveredDeviceVlans ( long policyId, long deviceId )
```

Parameter	Description
policyId	The object ID for the IPv4 reconciliation policy.
deviceId	The object ID of the discovered device.

Get Discovered Device ARP Entries

Returns all ARP entries of a specific device discovered by running an IPv4 reconciliation policy.

Output / Response

Returns all ARP entries of a specific device.

To understand the returned result, refer to the constants listed in [Device Properties](#) on page 190.

API call:

```
APIEntity[] getDiscoveredDeviceArpEntries ( long policyId, long deviceId )
```

Parameter	Description
policyId	The object ID for the IPv4 reconciliation policy.
deviceId	The object ID of the discovered device.

Get Discovered Device MAC Address Entries

Returns all MAC address entries of a specific device discovered by running an IPv4 reconciliation policy.

Output / Response

Returns all MAC address entries of a specific device.

To understand the returned result, refer to the constants listed in [Device Properties](#) on page 190.

API call:

```
APIEntity[] getDiscoveredDeviceMacAddressEntries ( long policyId, long deviceId )
```

Parameter	Description
policyId	The object ID for the IPv4 reconciliation policy.
deviceId	The object ID of the discovered device.

IPv6 Objects

IPv6 objects include blocks, networks, DHCP ranges, addresses, DUID and reconciliation policy.

Add IPv6 Address

Adds an IPv6 address to a specified IPv6 network.

Output / Response

Returns the object ID of the new IPv6 address..

API call:

```
long addIPv6Address ( long containerId, String address, String type, String name, String properties )
```

Parameter	Description
containerId	The object ID of the container in which the IPv6 address is being added. This can be the object ID of a Configuration, IPv6 block or IPv6 network. The parent IPv6 network object must exist before adding an IPv6 address, otherwise an error will occur.
address	The IPv6 address to be added. This value cannot be empty. address and type must be consistent. For example, if the type is ObjectTypes.IP6Address, the address must be a string representing an IPv6 address.
type	The type of IPv6 address. This value must be one of the following: macAddress , IP6Address , or InterfaceID . address and type must be consistent. For example, if the type is ObjectTypes.IP6Address, the address must be a string representing an IPv6 address.
name	Descriptive name for the IPv6 address. This value can be empty.
properties	Adds object properties, including user-defined fields.

Add IPv6 Block by MAC Address

Adds a IPv6 block by specifying the MAC address of the server.

Output / Response

Returns the object ID of the new IPv6 block.

API call:

```
long addIPv6BlockByMACAddress ( long parentId, String macAddress, String name, String properties )
```

Parameter	Description
parentId	The object ID of the parent object of the new IPv6 block. The parent object must be another IPv6 block.
macAddress	The MAC address of the server in the format <i>nnnnnnnnnnnn</i> , <i>nn-nn-nn-nn-nn-nn</i> or <i>nn:nn:nn:nn:nn:nn</i> , where <i>nn</i> is a hexadecimal value.
name	Descriptive name for the IPv6 block. This value can be empty.
properties	Adds object properties, including user-defined fields. This value can be empty.

Add IPv6 Block by Prefix

adds an IPv6 block by specifying the prefix for the block.

Output / Response

Returns the object ID of the new IPv6 block.

API call:

```
long addIPv6BlockByPrefix ( long parentId, String prefix, String name, String properties )
```

Parameter	Description
parentId	The object ID of the parent object of the new IPv6 block. The parent object may be a configuration or another IPv6 block.
prefix	The IPv6 prefix for the new block. This value cannot be empty.
name	Descriptive name for the IPv6 block. This value can be empty.
properties	Adds object properties, including user-defined fields. This value can be empty.

Add IPv6 Network by Prefix

adds an IPv6 network by specifying the prefix for the network.

Output / Response

Returns the object ID of the new IPv6 network.

API call:

```
long addIPv6NetworkByPrefix ( long parentId, String prefix, String name, String properties )
```

Parameter	Description
parentId	The object ID of the IPv6 block in which the new IPv6 network will be located.
prefix	The IPv6 prefix for the new network. This value cannot be empty.
name	Descriptive name for the IPv6 network. This value can be empty.

Parameter	Description
properties	Adds object properties, including user-defined fields. This value can be empty.

Split IPv6 Block or Network

Splits an IPv6 block or network into the specified number of blocks or networks.

Output / Response

Returns an array of IPv6 blocks or networks created after splitting the block or network.

API call:

```
APIEntity[] splitIPv6Range ( long rangeId, int numberOfParts, String options )
```

Parameter	Description
rangeId	The database object ID of the block or network that is being split.
numberOfParts	The number of the blocks or networks into which the block or network will be split. Valid values are 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024.
options	No options available. Reserved for future use.

Get IPv6 Range by IP Address

Returns the DHCPv6 Range containing the specified IPv6 address. Use this method to find the Configuration, IPv6 Block, IPv6 Network, or DHCPv6 Range containing a specified address. You can specify the type of object to be returned, or you can leave the type of object empty to find the most direct container for the object..

Output / Response

Returns an APIEntity for the object containing the specified address. If no object is found, returns an empty APIEntity. If **ObjectTypes.IP6Block**, **ObjectTypes.IP6Network**, or **ObjectTypes.DHCP6Range** is specified as the *type* parameter, returns an object of the specified type. If an empty string is specified as the *type* parameter, returns the most direct container for the IPv6 address.

API call:

```
APIEntity getIPRangedByIP ( long containerId, String type, String address )
```

Parameter	Description
containerId	The object ID of the container in which the IPv6 address is located. This can be a Configuration, IPv6 Block, IPv6 Network, or DHCPv6 Range. When you do not know the block, network, or range in which the address is located, specify the configuration.
type	The type of object containing the IPv6 address. Specify ObjectTypes.IP6Block , ObjectTypes.IP6Network , or ObjectTypes.DHCP6Range to find the block, network, or range containing the IPv6 address. Specify an <i>empty string</i> to return the most direct container for the IPv6 address.
address	An IPv6 address.

Get IPv6 Objects by Hint


Returns an array of IPv6 objects found under a given container object. The networks can be filtered by using **ObjectProperties.hint** and **ObjectProperties.accessRight** options. Currently, it only supports IPv6 networks.

Output / Response

Returns an array of IPv6 objects based on the input argument without their properties fields populated, or returns an empty array if *containerId* is invalid. If no access right option is specified, the View access level will be used by default.

API call:

```
APIEntity[] getIP6ObjectsByHint ( long containerId, String objectType, int start, int count, String options )
```

Parameter	Description
containerId	The object ID for the container object. It can be the object ID of any object in the parent object hierarchy. The highest parent object is the configuration level.
objectType	The type of object containing the IPv6 network. Currently, it only supports ObjectTypes.IP6Network .
start	Indicates where in the list of objects to start returning objects. The list begins at an index of 0.
count	Indicates the maximum number of child objects that this method will return.
options	<p>A string containing options. The Option names available in <i>ObjectProperties</i> are ObjectProperties.hint and ObjectProperties.accessRight.</p> <p>Multiple options can be separated by a (pipe) character. For example:</p> <pre>hint=ab accessRight=ADD</pre> <p>The values for the ObjectProperties.hint option can be the prefix of the IP address for a network or the name of a network.</p> <ul style="list-style-type: none"> Example 1 <p>The following example will match networks that have the network ID starting with 2000::. For example, 2000::/64.</p> <pre>String options = ObjectProperties.hint + "=2000::"</pre> Example 2 <p>The following example will match networks that have a name starting with "abc". For example, "abc", "abc123" or "abcdef".</p> <pre>String options = ObjectProperties.hint + "=abc "</pre> <p> Note: Matching networks to a network ID (Example 1) will take precedence over matching networks to a name (Example 2).</p> <p>The values for the ObjectProperties.accessRight option must be one of the constants listed in Access Right Values on page 189 and Object Types on page 209. For example:</p> <pre>String options = ObjectProperties.accessRight + "=" + AccessRightValues.AddAccess;</pre>

Assign IPv6 Address

Assigns an IPv6 address to a MAC address and host.


Output / Response

Returns true if the IPv6 address is successfully assigned; returns false if the address is not successfully assigned.

API call:

```
boolean assignIPv6Address ( long containerId, String address, String action, String
macAddress, String hostInfo, String properties )
```

Parameter	Description
containerId	The object ID of the container in which the IPv6 address is being assigned. This can be the object ID of a Configuration, IPv6 block or IPv6 network. The parent IPv6 network object must exist before adding an IPv6 address, otherwise an error will occur.
address	The IPv6 address to be assigned. This value cannot be empty. The address must be created with <code>addIPv6Address()</code> before it can be assigned. For more information, refer to Add IPv6 Address on page 85. The address must be a string representing an IPv6 address.
action	Determines how to assign the address. Valid values are MAKE_STATIC or MAKE_DHCP_RESERVED .
macAddress	The MAC address in the format <i>nnnnnnnnnnnnnn</i> , <i>nn-nn-nn-nn-nn-nn</i> or <i>nn:nn:nn:nn:nn:nn</i> , where <i>nn</i> is a hexadecimal value.
hostInfo	<p>The host information for the IPv6 address. This value can be empty. The <code>hostInfo</code> string uses the following format: <code>viewId, hostname, ifSameAsZone, ifReverseMapping</code>.</p> <p>Where viewId is the object ID of the DNS view; hostname is the name of DNS zone for the address; ifSameAsZone is a Boolean value, with <i>true</i> indicating that the name of the resource record should be the same as the host; ifReverseMapping is a Boolean value, with <i>true</i> indicating that a reverse record should be created.</p> <p>Shown here is an example of a hostInfo string: <code>2030445,www.example.com,false,true</code>.</p>
properties	<p>A string containing the following property, including user-defined fields:</p> <ul style="list-style-type: none"> ptrs—a string containing the list of unmanaged external host records to be associated with the IPv6 address in the following format: <pre>viewId,exHostFQDN[, viewId,exHostFQDN,...]</pre> <p>You can assign External Host records to an IPv6 address using the following method:</p> <pre>EntityProperties props = new EntityProperties(); props.addProperty(ObjectProperties.ptrs, "123,exHostFQDN.com,456,exHostFQDN.net") long addressId = service.assignIPv4Address(configurationId, IPv6Address, macAddressStr, hostInfo, IPAssignmentActionValues.MAKE_STATIC, props.getPropertiesString());</pre> name—name of the IPv6 address. DUID—DHCPv6 unique identifier. locationCode—the hierarchical location code consists of a set of 1 to 3 alpha-numeric strings separated by a space. The first two characters indicate

Parameter	Description
	<p>a country, followed by next three characters which indicate a city in UN/LOCODE. New custom locations created under a UN/LOCODE city are appended to the end of the hierarchy. For example, CA TOR OF1 indicates: CA= Canada TOR=Toronto OF1=Office 1.</p> <p> Note: The code is case-sensitive. It must be all UPPER CASE letters. The county code and child location code should be alphanumeric strings.</p> <p>This value can be <i>empty</i>.</p>

Clear IPv6 Address

Clears a specified IPv6 address assignment.

Output / Response

Returns *true* to indicate that the IPv6 address has been cleared, or returns *false* if the operation was unsuccessful.

API call:

```
boolean clearIPv6Address ( long addressId )
```

Parameter	Description
addressId	The object ID of the IPv6 address to be deleted.

Get Entity by Prefix

Returns an APIEntity for the specified IPv6 block or network.

Output / Response

Returns an APIEntity for the specified IPv6 block or network. The APIEntity is empty if the block or network does not exist.

API call:

```
APIEntity getEntityByPrefix ( long containerId, String prefix, String type )
```

Parameter	Description
containerId	The object ID of higher-level parent object (IPv6 block or configuration) in which the IPv6 block or network is located.
prefix	The prefix value for the IPv6 block or network. This value cannot be empty.
type	The type of object to be returned. This value must be either IP6Block or IP6Network.

Get IPv6 Address

Returns an APIEntity for the specified IPv6 address.

Output / Response

Returns an APIEntity for the specified IPv6 address. The APIEntity is empty if the IPv6 address does not exist.

API call:

```
APIEntity getIPv6Address ( long containerId, String address )
```

Parameter	Description
containerId	The object ID of the container in which the IPv6 address is located. The container can be a configuration, an IPv6 block, or an IPv6 network.
address	The IPv6 address.

Reassign IPv6 Address

Reassigns an existing IPv6 address to a new IPv6 address. The destination address can be specified as an IPv6 address or as a MAC address from which an IPv6 address can be calculated.

Output / Response

Returns the object ID of the reassigned IPv6 address.

API call:

```
long reassignIP6Address ( long oldAddressId, String destination, String properties )
```

Parameter	Description
oldAddressId	The object ID of the current IPv6 address.
destination	The destination of the reassigned address. This can be specified as an IPv6 address string (NetworkID and InterfaceID), or as a MAC address from which the new IPv6 address can be calculated. The MAC address can be specified in the format <i>nnnnnnnnnnnn</i> or <i>nn-nn-nn-nn-nn-nn</i> , where <i>nn</i> is a hexadecimal value.
properties	Adds object properties, including user-defined fields.

Provision Devices

Provision and manage new devices for the network.

Add Device Instance

Used to provision new devices for the network and combines a number of existing API methods into one. This method assigns the next available, or manually defined, IP address and optionally adds a DNS host record and MAC address that are linked to the IP address and returns the property string containing IP address, netmask and gateway. When configured with a DNS host record, `addDeviceInstance()` will update the DNS server to immediately deploy the host record.



Note: Address Manager adds the DNS host record directly to the DNS/DHCP Server so that the individual host record is made live instantly. This is done through the Address Manager to DNS/DHCP Server communication service (Command Server) and does NOT require a standard Address Manager deployment.




Output / Response

Returns the property string containing IP address, netmask and gateway.

API call:

```
String addDeviceInstance ( String configName, String deviceName, String ipAddressMode, String ipEntity, String viewName, String zoneName, String recordName, String macAddressMode, String macEntity, String options )
```

Parameter	Description
configName	Name of parent configuration. If the value is empty or cannot be found, an exception will be thrown.

Parameter	Description
deviceName	IP/device name of the new instance. Reserved for future use.
ipAddressMode	Accepted values are REQUEST_STATIC , REQUEST_DHCP_RESERVED and PASS_VALUE . REQUEST_STATIC or REQUEST_DHCP_RESERVED is used to get the next available IP address. PASS_VALUE is used to pass an existing IP address. Metadata values will be updated to the newly assigned IP address only. REQUEST_DHCP_RESERVED is reserved for future use.
ipEntity	If <i>ipAddressMode</i> is REQUEST_STATIC or REQUEST_DHCP_RESERVED , this needs to be the network where the IP address will be provisioned from in the format of an IP address range in CIDR format or range. If <i>ipAddressMode</i> is PASS_VALUE , this needs to be an IP address.
viewName	Name of parent view.  Note: Specify an <i>empty string</i> ("") for all <i>viewName</i> , <i>zoneName</i> and <i>recordName</i> parameters to ignore DNS object creation.
zoneName	Parent zone of the record. This must be specified and existing if the <i>viewName</i> parameter is an empty string and existing.  Note: Specify an <i>empty string</i> ("") for all <i>viewName</i> , <i>zoneName</i> and <i>recordName</i> parameters to ignore DNS object creation.
recordName	Name of the host record to add. This cannot be empty if both viewName and zoneName are specified and in use. The viewName , zoneName and recordName parameters need to be used together: the values must <i>all</i> be an <i>empty string</i> (""), or they must all be populated with specific values.  Note: If all three parameter values are an empty string, DNS objects will not be created but an IP address will be assigned from a network and linked to a MAC address.
macAddressMode	Accepted values are REQUEST_VALUE or PASS_VALUE . If specified an <i>empty string</i> (""), the <i>MACEntity</i> parameter will be ignored. Use PASS_VALUE to manually provide the MAC address linked to IP address. REQUEST_VALUE is reserved for future use.
macEntity	If <i>macAddressMode</i> is PASS_VALUE , this must be a MAC address. If <i>macAddressMode</i> is REQUEST_VALUE , this is a MAC mask.
options	The options string contains four properties; skip , offset , excludeDHCPRange and allowDuplicateHosts . <ul style="list-style-type: none"> • skip—This is optional. It is applied to REQUEST_STATIC and REQUEST_DHCP_RESERVED for <i>ipAddressMode</i>. It is used to specify the IP address ranges or IP addresses to skip, separated by comma. A hyphen(-), not a dash is used to separate the start and end addresses. • offset—This is optional. It is applied to REQUEST_VALUE for <i>ipAddressMode</i>. This is to specify from which address to start to assign IPv4 Address. • excludeDHCPRange—This specifies whether IP addresses in DHCP ranges should be excluded from assignment or not. It is applied to REQUEST_STATIC only for <i>ipAddressMode</i>. The value is either true or false, default value is false. The value will always be set to true if the <i>ipAddressMode</i> is REQUEST_DHCP_RESERVED.

Parameter	Description
	<ul style="list-style-type: none"> • allowDuplicateHosts—This specifies whether the IP address can be added to an existing host record or not. The value is either true or false, default value is false. <p>➔ Note: The values for skip and offset must be IPv4 addresses and must appear in dotted octet notation.</p>

Delete Device Instance

Deletes either the IP address or MAC address (and all related DNS entries including host records, PTR records, or DHCP reserved addresses) on both the Address Manager and DNS/DHCP Server based on the IPv4 address or a MAC address supplied.

- ➔ **Note:** Address Manager adds the DNS host record directly to the DNS/DHCP Server so that the individual host record is made live instantly. This is done through the Address Manager to DNS/DHCP Server communication service (Command Server) and does NOT require a standard Address Manager deployment.

Output / Response

None.

API call:

```
void deleteDeviceInstance ( String configName, String identifier, String options )
```

Parameter	Description
configName	Name of parent configuration. If the value is empty or cannot be found, an exception will be thrown.
identifier	IP address or MAC address. If the value is empty or cannot be found, an exception will be thrown. Relevant IP addresses, host records and MAC addresses linked to multiple entities will be updated. Relevant IP addresses, host records and MAC addresses linked to a single entity will be deleted.
options	Currently empty. This parameter is reserved for future use.

DHCP

DHCP is an essential part of IPAM. DHCP manages the dynamic allocation of IP addresses on an IP network using the concept of address leases.

In Address Manager, DHCP is integrated into the IP core and defined using range objects. At most levels of the IP core, deployment options control the behavior of the DHCP service for an object and its descendant objects within the Address Manager managed network. A deployment role can also be associated with a server to provide DHCP services for a specific subnet.

IPv4 DHCP Ranges

DHCP ranges indicate the portion of a network that is dedicated to DHCP.

Ranges can have deployment options assigned to them to control the exact settings that clients receive. Address Manager then manages the deployment of the DHCP ranges to the managed server and activates the configuration.

Add IPv4 DHCP Range

Adds IPv4 DHCP ranges.

Output / Response

Returns the object ID for the new DHCPv4 range.

API call:

```
long addDHCP4Range ( long networkId, String start, String end, String properties )
```

Parameter	Description
networkId	The object ID for the network in which this DHCP range is located.
start	An IP address defining the lowest address or start of the range.
end	An IP address defining the highest address or end of the range.
properties	Adds object properties, including the object name and user-defined fields.

Add IPv4 DHCP Range By Size


Adds IPv4 DHCP ranges by offset and percentage.

Output / Response

Returns the object ID for the new DHCPv4 range.

API call:

```
long addDHCP4RangeBySize ( long networkId, String offset, String size, String properties )
```

Parameter	Description
networkId	The object ID for the network in which this DHCP range is located.
offset	An integer value specifying the point where the range should begin. The positive values indicate that the starting IP address of the range will be counted from the Network ID (first IP address) and forward in the range. The negative values indicate that the starting IP address of the range will be counted from the Network Broadcast Address (last IP address) and backward in the range.
size	The size of the range. Currently the range size can only be specified in a relative size in proportion to the parent network size. To define the relative range size, <i>defineRangeBy</i> must be set with the OFFSET_AND_PERCENTAGE value in the properties field.
properties	Optional object properties that can contain the object name, the value of <i>defineRangeBy</i> , and user-defined fields. The possible values for <i>defineRangeBy</i> are OFFSET_AND_SIZE and OFFSET_AND_PERCENTAGE .  Note: OFFSET_AND_SIZE is reserved for future use.

Get IPv4 Range by IP Address

Returns the DHCP Range containing the specified IPv4 address. Use this method to find the Configuration, IPv4 Block, IPv4 Network, or DHCP Range containing a specified address. You can specify the type of object to be returned, or you can leave the type of object empty to find the most direct container for the object.

Output / Response

Returns an `APIEntity` for the object containing the specified address. If no object is found, returns an *empty APIEntity*. If **ObjectTypes.IP4Block**, **ObjectTypes.IP4Network**, or **ObjectTypes.DHCP4Range** is specified as the *type* parameter, returns an object of the specified type. If an *empty string* ("") is specified as the *type* parameter, returns the most direct container for the IPv4 address.

API call:

`APIEntity getIPRangedByIP (long containerId, String type, String address)`

Parameter	Description
containerId	The object ID of the container in which the IPv4 address is located. This can be a Configuration, IPv4 Block, IPv4 Network, or DHCP Range. When you do not know the block, network, or range in which the address is located, specify the configuration.
type	The type of object containing the IPv4 address. Specify ObjectTypes.IP4Block , ObjectTypes.IP4Network , or ObjectTypes.DHCP4Range to find the block, network, or range containing the IPv4 address. Specify an <i>empty string</i> ("") to return the most direct container for the IPv4 address.
address	An IPv4 address.

Get IPv4 DHCP Range

Returns an IPv4 DHCP range object by calling it using its range.

Output / Response

Returns the specified DHCP4Range object from the database.

API call:

`APIEntity getEntityByRange (long parentId, String address1, String address2, String type)`

Parameter	Description
containerId	The object ID of the parent object of the DHCP range.
address1	An IP address defining the lowest address or start of the range.
address2	An IP address defining the highest address or end of the range.
type	The type of object returned: DHCP4Range. This must be one of the constants listed in Object Types on page 209.

Get IPv4 DHCP Ranges

Returns multiple IPv4 DHCP ranges for the specified parent ID.

Output Response

Returns an array of DHCPv4 range objects from the database.

API call:

`APIEntity[] getEntities (long parentId, String type, int start, int count)`

Parameter	Description
parentId	The object ID of the parent object of the DHCP range.
type	The type of object returned: DHCP4Range. This must be one of the constants listed in Object Types on page 209.

Parameter	Description
start	Indicates where in the list of child ranges to start returning range objects. The list begins at an index of 0.
count	The type of object returned: DHCP4Range. This must be one of the constants listed in Object Types on page 209.

Get Max Allowed Range

Finds the maximum possible address range to which the existing IPv4 DHCP range can be extended. This method only supports the IPv4 DHCP range.

Output / Response

Returns the possible start address and end address for the specified IPv4 DHCP range object in the form of array of length 2.

API call:

```
String[] getMaxAllowedRange ( long rangeId )
```

Parameter	Description
rangeId	The object ID of the IPv4 DHCP range.

Update IPv4 DHCP Range

A DHCP range's **name** property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

IPv4 DHCP Range Generic Methods

DHCP ranges can be deleted using the generic `delete()` method.

For more information, see [Deleting Objects](#) on page 52.

IPv6 DHCP Ranges

DHCPv6 ranges indicate the portion of a network that is dedicated to DHCPv6.

Ranges can have deployment options assigned to them to control the exact settings that clients receive. Address Manager then manages the deployment of the DHCPv6 ranges to the managed server and activates the configuration.

Add IPv6 DHCP Range

Adds IPv6 DHCP ranges.

Output / Response

Returns the object ID for the new DHCPv6 range.

API call:

```
long addDHCP6Range ( long networkId, String start, String end, String properties )
```

Parameter	Description
networkId	The object ID for the network in which this DHCPv6 range is located.
start	An IP address defining the lowest address or start of the range.

Parameter	Description
end	An IP address defining the highest address or end of the range.
properties	Adds object properties, including the object name and user-defined fields.

Add IPv6 DHCP Range By Size

Adds IPv6 DHCP ranges by size.

Output / Response

Returns the object ID for the new DHCPv6 range.

API call:

```
long addDHCP6RangeBySize ( long networkId, String start, String size, String properties )
```

Parameter	Description
networkId	The object ID for the network in which this DHCP range is located.
start	<p>An integer value or IPv6 address specifying the point where the range should begin. The positive values indicate that the starting IP address of the range will be counted from the Network ID (first IP address) and forward in the range. The IPv6 address value indicates the starting address of the range.</p> <p>➔ Note:</p> <ul style="list-style-type: none"> If <i>defineRangeBy</i> is set with the AUTOCREATE_BY_SIZE value in the property field, the start field must contain an empty string (""). If <i>defineRangeBy</i> is set with the OFFSET_AND_SIZE value in the property field, the start field must contain a non-negative integer value. If <i>defineRangeBy</i> is set with the START_ADDRESS_AND_SIZE value in the property field, the start field must contain a valid IPv6 address that will be used as the starting address of the DHCP range.
size	The size of the range. Currently the range size can only be specified in a relative size in proportion to the parent network size.
properties	<p>Optional object properties that can contain the object name, the value of <i>defineRangeBy</i>, and user-defined fields. The possible values for <i>defineRangeBy</i> are AUTOCREATE_BY_SIZE, OFFSET_AND_SIZE, and START_ADDRESS_AND_SIZE.</p> <p>➔ Note: If you do not specify the <i>defineRangeBy</i> value, the DHCP range will be created using AUTOCREATE_BY_SIZE by default.</p>

Get IPv6 Range by IP Address

Returns the DHCPv6 Range containing the specified IPv6 address.

Use this method to find the Configuration, IPv6 Block, IPv6 Network, or DHCPv6 Range containing a specified address. You can specify the type of object to be returned, or you can leave the type of object empty ("") to find the most direct container for the object.

Output / Response

Returns an *APIEntity* for the DHCPv6 Range containing the specified IPv6 address. If no object is found, returns an *empty APIEntity*. If **ObjectTypes.IP6Block**, **ObjectTypes.IP6Network**, or **ObjectTypes.DHCP6Range** is specified as the *type* parameter, returns an object of the specified type.

If an *empty string* ("") is specified as the *type* parameter, returns the most direct container for the IPv6 address.

API call:

`APIEntity getIPRangedByIP (long containerId, String type, String address)`

Parameter	Description
containerId	The object ID of the container in which the IPv6 address is located. This can be a Configuration, IPv6 Block, IPv6 Network, or DHCP Range. When you do not know the block, network, or range in which the address is located, specify the configuration.
type	The type of object containing the IPv6 address. Specify ObjectTypes.IP6Block , ObjectTypes.IP6Network , or ObjectTypes.DHCP6Range to find the block, network, or range containing the IPv6 address. Specify an <i>empty string</i> ("") to return the most direct container for the IPv6 address.
address	An IPv6 address.

Get IPv6 DHCP Range

Returns an IPv6 DHCP range by calling it using its range.

Output / Response

Returns the specified DHCP6Range object from the database.

API call:

`APIEntity getEntityByRange (long parentId, String address1, String address2, String type)`

Parameter	Description
parentId	The object ID of the parent object of the DHCPv6 range.
address1	An IP address defining the lowest address or start of the range.
address2	An IP address defining the highest address or end of the range.
type	The type of object returned: DHCPv6 Range. This must be one of the constants listed in Object Types on page 209.

Get Multiple IPv6 DHCP Ranges

Returns multiple DHCPv6 range objects for the specified parent ID.

Output / Response

Returns an array of DHCPv6 range objects from the database.

API call:

`APIEntity[] getEntities (long parentId, String type, int start, int count)`

Parameter	Description
parentId	The object ID of the parent object of the DHCPv6 range.
type	The type of object returned: DHCPv6 Range. This must be one of the constants listed in Object Types on page 229.
start	Indicates where in the list of child ranges to start returning range objects. The list begins at an index of 0.

Parameter	Description
count	The maximum number of DHCPv6 range objects to return.

Update IPv6 DHCP Range

A DHCPv6 range's name property can be updated using the generic `update()` method.

For more information, refer to [Updating Objects](#) on page 50.

IPv6 DHCP Range Generic Methods

DHCPv6 ranges can be deleted using the generic `delete()` method.

For more information, refer to [Deleting Objects](#) on page 52.

DHCP Client Options

DHCP options that can be added to a DHCP configuration to specify deployment instructions relating to extra settings for client configuration.

For more information about these options, refer to RFCs 2132, 2242, 2610, 2241, and 2485. Readers are also encouraged to examine RFCs 1497 and 1122 for background information. Options that accept Boolean values are activated by a value of 1 unless otherwise specified. When specifying a list of IPv4 addresses, the first address takes precedence.

Add DHCP Client Option

Adds DHCP client options and returns the object ID for the new option object.

Output / Response

Returns the object ID for the new DHCPv4 client option object.

API call:

```
long addDHCPClientDeploymentOption ( long entityId, String name, String value, String properties )
```

Parameter	Description
entityId	The object ID for the entity to which the deployment option is being added.
name	The name of the DHCPv4 client option being added. This name must be one of the constants listed in DHCP Client Options on page 191.
value	The value being assigned to the option.
properties	Adds object properties, including user-defined fields.

Get DHCP Client Option

Returns DHCPv4 client options assigned for the object specified excluding the options inherited from the higher-level parent object.

Output / Response

Returns the specified DHCPv4 client option object from the database.

API call:

```
APIDeploymentOption getDHCPClientDeploymentOption ( long entityId, String name, long serverId )
```

Parameter	Description
entityId	The object ID for the entity to which the deployment option has been applied.
name	The name of the DHCPv4 client option being added. This name must be one of the constants listed in DHCP Client Options on page 191.
serverId	The specific server or server group to which this option is deployed. To return an option that has not been assigned to a server, set this value to 0(zero). Omitting this parameter from the method call will result in an error.

Update DHCP Client Option


Updates DHCP client options.

Output / Response

None.

API call:

```
void updateDHCPClientDeploymentOption ( APIDeploymentOption option )
```

Parameter	Description
option	The DHCP client option object to be updated.  Note: The Name field of the DHCP client deployment option object cannot be updated.

Delete DHCP Client Option

Deletes DHCP client options.

Output / Response

None.

API call:

```
void deleteDHCPClientDeploymentOption ( long entityId, String name, long serverId )
```

Parameter	Description
entityId	The object ID for the entity from which the deployment option will be deleted.
name	The name of the DHCPv4 client option to be deleted. This name must be one of the constants listed in DHCP Client Options on page 191.
serverId	The specific server or server group to which this option is deployed. To delete an option that has not been assigned to a server, set this value to 0 (zero). Omitting this parameter from the method call will result in an error.

DHCP6 Client Options

DHCPv6 options that can be added to a DHCP configuration to specify deployment instructions relating to extra settings for client configuration.

Add DHCP6 Client Option

Adds DHCPv6 client options and returns the database object ID for the new option object.

Output / Response

Returns the object ID of the new DHCPv6 client object.

API call:

```
long addDHCP6ClientDeploymentOption ( long entityId, String name, String value, String properties )
```

Parameter	Description
entityId	The object ID for the entity to which the deployment option is being added.
name	The name of the DHCPv6 client option being added. This name must be one of the constants listed in the DHCP6 Client Options on page 194.
value	The value being assigned to the option.
properties	Adds object properties, including user-defined fields.

Get DHCP6 Client Option

Returns DHCP6 client options assigned for the object specified excluding the options inherited from the higher-level parent object.

Output / Response

Returns the specified DHCPv6 client option object from the database.

API call:

```
APIDeploymentOption getDHCP6ClientDeploymentOption ( long entityId, String name, long serverId )
```

Parameter	Description
entityId	The object ID for the entity.
name	The name of the DHCPv6 client option being added. This name must be one of the constants listed in the DHCP6 Client Options on page 194.
serverId	The specific server or server group to which this option is deployed. To return an option that has not been assigned to a server role, set this value to 0 (zero) . Omitting this parameter from the method call will result in an error.

Update DHCP6 Client Option


Updates DHCPv6 client options.

Output / Response

None.

API call:

```
void updateDHCP6ClientDeploymentOption ( APIDeploymentOption option )
```

Parameter	Description
entityId	The DHCPv6 client option object that is updated.  Note: The Name field of the DHCPv6 client deployment option object cannot be updated.

Delete DHCP6 Client Option

Deletes DHCPv6 client options.

Output / Response

None.

API call:

```
void deleteDHCP6ClientDeploymentOption ( long entityId, String name, long serverId )
```

Parameter	Description
entityId	The database object ID for the entity from which this deployment option will be deleted.
name	The name of the DHCPv6 client option being deleted. This name must be one of the constants listed in the DHCP6 Client Options on page 194.
serverId	The specific server or server group to which this option is deployed. To delete an option that has not been assigned to a server role, set this value to 0 (zero) . Omitting this parameter from the method call will result in an error.

DHCP Custom Options

This method is available to add DHCP custom option definitions.

Add Custom Deployment Option

Adds a custom deployment option.

Output / Response

Returns the object ID of the new option defined.

API call:

```
long addCustomOptionDefinition ( long configurationId, String name, long optionId, String optionType, boolean allowMultiple, String properties )
```

Parameter	Description
configurationId	The object ID of the parent configuration.
name	The name of the custom deployment option. This value cannot be empty.
optionId	The option code for the custom deployment option. This value must be within the range of 151 to 174, 178 to 207, 212 to 219, 222 to 223, or 224 to 254.
optionType	The type of custom deployment option. This value must be one of the items listed in DHCP Custom Option Types on page 194.
allowMultiple	<p>Determines whether or not the custom option requires multiple values. The default value is false. The value cannot be empty.</p> <p>In Perl script, only an empty string and 0 (zero) are considered as false; other values are considered as true. Therefore, a string containing the word <i>false</i> is considered to be true because the string is not empty. In Perl, set the <i>allowMultiple</i> data type to <i>string</i> and set the value to either true or false:</p> <pre>SOAP::Data->type('string')-> name('allowMultiple')-> value("false")-> attr({xmlns => ''})</pre>

Parameter	Description
	<p>Or, set the <i>allowMultiple</i> data type to <i>boolean</i>. Set the value to either 0 or an empty string to represent false. Set the value to any other text to represent true.</p> <pre>SOAP::Data->type('boolean') ->name('allowMultiple') ->value(0) ->attr({xmlns => ''})</pre>
properties	Adds object properties, including user-defined fields

DHCP Service Options

These are the DHCP service options that can be added to a DHCP configuration to specify deployment instructions relating to extra settings for service configuration.

Add DHCP Service Option

Adds DHCP service options.

Output / Response

Returns the object ID for the new DHCPv4 service option.

API call:

```
long addDHCPServiceDeploymentOption ( long entityId, String name, String value, String
properties )
```

Parameter	Description
entityId	The object ID for the entity to which the deployment option is being added.
name	The name of the DHCPv4 service option being added. This name must be one of the constants listed in DHCP Service Options on page 195.
value	<p>The value being assigned to the option.</p> <p>When adding the DDNS hostname option, you need to specify the value in the following format: [Type],[Position],[Data] for IP and MAC type, and [Type],[Data] for FIXED type. Where:</p> <ul style="list-style-type: none"> Type—type of DDNS hostname. The possible values are <code>DHCPServiceOptionConstants.DDNS_HOSTNAME_TYPE_IP</code>, <code>DHCPServiceOptionConstants.DDNS_HOSTNAME_TYPE_MAC</code>, or <code>DHCPServiceOptionConstants.DDNS_HOSTNAME_TYPE_FIXED</code>. Position—specify where you wish to add the data value to the IP or MAC address. The possible values are <code>DHCPServiceOptionConstants.DDNS_HOSTNAME_POSITION_PREPEND</code>, or <code>DHCPServiceOptionConstants.DDNS_HOSTNAME_POSITION_APPEND</code>. This is only required for IP or MAC type with Data. Data—For IP and MAC address, this value is used to be prepended or appended to the IP address or MAC address. For FIXED type, this value must be specified and will be used for the DDNS hostname. This is optional for IP and MAC type.
properties	Adds object properties, including user-defined fields.

Get DHCP Service Option

Returns DHCP service options assigned for the object specified excluding the options inherited from the higher-level parent object.

Output / Response

Returns the requested DHCPv4 service option object from the database.

API call:

```
APIDeploymentOption getDHCPServiceDeploymentOption ( long entityId, String name, long serverId )
```

Parameter	Description
entityId	The object ID for the entity to which the deployment option is assigned.
name	The name of the DHCPv4 service option being retrieved. This name must be one of the constants listed in DHCP Service Options on page 195.
serverId	Specifies the server or server group to which the option is deployed for the specified entity. To retrieve an option that has not been assigned to a server role, specify 0 as a value. Omitting this parameter from the method call will result in an error.

Update DHCP Service Option

Updates DHCP service options.



Note: The Name field of the DHCP service deployment option object cannot be updated.

Output / Response

None.

API call:

```
void updateDHCPServiceDeploymentOption ( APIDeploymentOption option )
```

Parameter	Description
option	The DHCP service option object to be updated.

Delete DHCP Service Option

Deletes DHCP service options.



Note: The Name field of the DHCP service deployment option object cannot be updated.

Output / Response

None.

API call:

```
void deleteDHCPServiceDeploymentOption ( long entityId, String name, long serverId )
```

Parameter	Description
entityId	The object ID for the entity from which this deployment option is being deleted.
name	The name of the DHCPv4 service option being deleted. This name must be one of the constants listed in DHCP Service Options on page 212.

Parameter	Description
serverId	Specifies the server or server group to which the option is deployed for the specified entity. To retrieve an option that has not been assigned to a server role, set this value to 0(zero). Omitting this parameter from the method call will result in an error.

DHCP6 Service Options

These are the DHCPv6 service options that can be added to a DHCP configuration to specify deployment instructions relating to extra settings for service configuration.

Add DHCP6 Service Option

Adds DHCPv6 service options.

Output / Response

Returns the object ID for the new option.

API call:

```
long addDHCP6ServiceDeploymentOption ( long entityId, String name, String value, String properties )
```

Parameter	Description
entityId	The object ID for the entity to which the deployment option is being added.
name	The name of the DHCPv6 service option being added. This name must be one of the constants listed in DHCP6 Service Options on page 196.
value	<p>The value being assigned to the option.</p> <p>When adding the DDNS hostname option, you need to specify the value in the following format: [Type],[Data] for IP and DUID. Where:</p> <ul style="list-style-type: none"> Type—type of DDNS hostname. The possible values are DHCPServiceOptionConstants.DDNS_HOSTNAME_TYPE_IP, DHCPServiceOptionConstants.DDNS_HOSTNAME_TYPE_DUID. Data—For IP and DUID, this value is used to form the DDNS hostname. This is optional.
properties	Adds object properties, including user-defined fields.

Get DHCP6 Service Option

Returns DHCPv6 service options assigned for the object specified excluding the options inherited from the higher-level parent object.

Output / Response

Returns the requested DHCPv6 service option object from the database.

API call:

```
APIDeploymentOption getDHCP6ServiceDeploymentOption ( long entityId, String name, long serverId )
```

Parameter	Description
entityId	The database object ID for the entity to which the deployment option is assigned.

Parameter	Description
name	The name of the DHCPv6 service option being added. This name must be one of the constants listed in DHCP6 Service Options on page 196.
serverId	Specifies the server or server group to which the option is deployed for the specified entity. To retrieve an option that has not been assigned to a server role, set this value to 0(zero). Omitting this parameter from the method call will result in an error.

Update DHCP6 Service Option

Updates DHCPv6 service options.



Note: The Name field of the DHCPv6 service deployment option object cannot be updated.

Output / Response

None.

API call:

```
void updateDHCP6ServiceDeploymentOption ( APIDeploymentOption option )
```

Parameter	Description
option	The DHCPv6 service option object to be updated.

Delete DHCP6 Service Option

Deletes DHCPv6 service options.

Output / Response

None.

API call:

```
void deleteDHCP6ServiceDeploymentOption ( long entityId, String name, long serverId )
```

Parameter	Description
entityId	The object ID for the entity from which this deployment option is being deleted.
name	The name of the DHCPv4 service option being deleted. This name must be one of the constants listed in DHCP6 Service Options on page 196.
serverId	Specifies the server or server group to which the option is deployed for the specified entity. To return an option that has not been assigned to a server role, set this value to 0(zero). Omitting this parameter from the method call will result in an error.

DHCP Vendor Profiles and Options

Vendor profiles are sets of DHCP options required by particular devices. For example, a VoIP phone might need a very specific set of DHCP options.

A vendor profile is the container in which the DHCP vendor options are stored. A single vendor profile contains several DHCP vendor options, all of which can be sent to a specific type of device such as an IP handset. After the vendor profile is created and populated with the appropriate options, the options themselves can be defined and set at the appropriate level.

Add DHCP Vendor Deployment Option

Adds a DHCP vendor deployment to specified objects.

Output / Response

Returns the object ID of the new DHCP vendor deployment option.

API call:

```
long addDHCPVendorDeploymentOption ( long parentId, long optionId, String value, String properties )
```

Parameter	Description
parentId	The object ID of the parent object for the DHCP vendor deployment option. The parent object must not be a DNS object. Valid parent types are Configuration, IP4Block, IP4Network, IP4NetworkTemplate, IP4Addr, IP4DHCPRange, Server, MACAddr, and MACPool.
optionId	The object ID of the vendor option definition. All DHCP vendor client deployment options have a fixed option code of 60 and a unique option sub-code. The unique sub-code is set with the <i>optionId</i> value in the <code>addVendorOptionDefinition()</code> method.
value	The value for the option. This value cannot be empty. The value should be appropriate for its option type. For example, if the option type is IP4 and <code>allowMultiple</code> is set as true in the vendor option definition, then the value of the DHCP vendor client deployment option should be multiple IPv4 addresses in a comma-separated list.
properties	Adds object properties, including user-defined fields. This value can be empty. If the DHCP vendor client deployment option is intended for use with a specific server, the object ID of the server must be specified in the <i>properties</i> string.

Add Vendor Option Definition

Adds a vendor option definition to a vendor profile.

Output / Response

Returns the object ID of the new vendor option definition.

API call:

```
long addVendorOptionDefinition ( long vendorProfileId, long optionId, String name, String optionType, String description, boolean allowMultiple, String properties )
```

Parameter	Description
vendorProfileId	The object ID of the vendor profile.
optionId	The deployment option ID. This value must be within the range of 1 to 254.
name	The name of the vendor option. This value cannot be empty.
optionType	The option type. This value must be one of the types listed in Vendor Profile Option Types on page 216.
description	A description of the vendor option. This value cannot be empty.
allowMultiple	Determines whether or not the custom option requires multiple values. The default value is false. This value cannot be empty.

Parameter	Description
	<p>In Perl script, only an empty string and 0 (zero) are considered as false; other values are considered as true. Therefore, a string containing the word “false” is considered to be true because the string is not empty. In Perl, set the <i>allowMultipliedata</i> type to string and set the value to either true or false:</p> <pre>SOAP::Data->type('string')-> name('allowMultiple')-> value("false")-> attr({xmlns => ''})</pre> <p>Or, set the <i>allowMultipliedata</i> type to <i>boolean</i>. Set the value to either 0 or an empty string to represent false. Set the value to any other text to represent true.</p> <pre>SOAP::Data->type('boolean') ->name('allowMultiple') ->value(0) ->attr({xmlns => ''})</pre>
properties	Adds object properties, including user-defined fields. This value can be empty.

Add Vendor Profile

Adds a vendor profile and returns the object ID for the new vendor profile.

Output / Response

Returns the object ID of the new vendor profile.

API call:

```
long addVendorProfile ( String identifier, String name, String description, String
properties )
```

Parameter	Description
identifier	The Vendor Class Identifier.
name	A descriptive name for the vendor profile. This name is not matched against DHCP functionality.
description	A description of the vendor profile.
properties	Adds object properties, including user-defined fields.

Delete DHCP Vendor Deployment Option

Deletes a specified DHCP vendor deployment option.

Output / Response

None.

API call:

```
void deleteDHCPVendorDeploymentOption ( long entityId, long optionId, long serverId )
```

Parameter	Description
entityId	The object ID of the object to which the DHCP vendor deployment option is assigned.

Parameter	Description
optionId	the object ID of the vendor option definition. All DHCP vendor client deployment options have a fixed option code of 60 and a unique option sub-code. The unique sub-code is set with the <i>optionId</i> value in the <code>addVendorOptionDefinition()</code> method.
serverId	The object ID of the server or server group where the DHCP vendor deployment option is used. If the option is generic, set this value to 0 (zero). Omitting this parameter from the method call will result in an error.

Get DHCP Vendor Deployment Option

Retrieves a DHCP vendor deployment option assigned for the object specified excluding the options inherited from the higher-level parent object.

Output / Response

Returns an **APIDeploymentOption** for the DHCP vendor client deployment option. The **APIDeploymentOption** is *empty* if the specified option does not exist. The *property* string of the returned **APIDeploymentOption** contains at least contain the following substring to represent vendor option definitions:

```
optionId=optionID|optionType=Integer(-128 to 127) |
optionDescription=description
multipleSignedInt8|optionAllowMultiple=boolean|server=serverID|
```

server=serverID only appears if the DHCP vendor client deployment option is used for a specific server.

API call:

```
APIDeploymentOption getDHCPVendorDeploymentOption ( long entityId, long optionId, long serverId )
```

Parameter	Description
entityId	The object ID of the entity to which the DHCP vendor deployment option is assigned. This must be the ID of a configuration, IPv4 block, IPv4 network, IPv4 address, IPV4 DHCP rage, server, MAC address, or MAC Pool.
optionId	The object ID of the DHCP vendor option definition.
serverId	The specific server or server group to which this option is deployed for the specified entity. To return an option that has not been assigned to a server, set this value to 0 (zero). Omitting this parameter from the method call will result in an error.

Update DHCP Vendor Deployment Option

Updates the specified DHCP vendor deployment option.



Note: The Name field of the DHCP vendor deployment option object cannot be updated.

Output / Response

None.

API call:

```
void updateDHCPVendorDeploymentOption ( APIDeploymentOption option )
```

Parameter	Description
option	APIDeploymentOption to be updated. This is what getDHCPVendorDeploymentOption() returns.

DHCP Match Classes

Match classes in Address Manager (also known as classes in ISC DHCP and as user and vendor classes in Microsoft DHCP) allow you to restrict address allocation and assign options to clients that match specified criteria.

For example, using a match class, you could assign a specific DHCP lease length to clients that match a MAC address pattern or clients that are configured to send a specific identifier. A DHCP client becomes a member of a class when it matches the specified criteria.

Address Manager provides the following seven match criteria values:

- MATCH_HARDWARE
- MATCH_DHCP_CLIENT_ID
- MATCH_DHCP_VENDOR_ID
- MATCH_AGENT_CIRCUIT_ID
- MATCH_AGENT_REMOTE_ID
- CUSTOM_MATCH
- CUSTOM_MATCH_IF

Add DHCP Match Classes

Adds DHCP match classes to Address Manager.


Output / Response

Returns the object ID of the new DHCP match class added.

API Call:

```
long addDHCPMatchClass ( long configurationId, String name, String matchCriteria, String properties )
```

Parameter	Description
configurationId	The object ID of the configuration to which the DHCP match class is being added.
name	The name of the DHCP match class.
matchCriteria	A string defining the match criteria. The value must be one of the constants listed in DHCP Class Match Criteria on page 191.
properties	A string containing the following properties and values: <ul style="list-style-type: none"> • description—a description of the match class. • matchOffset—Match Offset value for the MatchClass. It refers to the point where the match should begin. • matchLength—Match Length value for the Matchclass. It refers to the number of characters to match. • customMatchRawString—a raw string that maps directly to a data or boolean expression for DHCP_CLASS_CUSTOM_MATCH and DHCP_CLASS_CUSTOM_MATCH_IF constants. Use the syntax and grammar supported by the ISC's DHCP daemon. End the string with a “,” semicolon. If you omit the semicolon, one is automatically added when the condition is deployed.

Parameter	Description
	 Note: <ul style="list-style-type: none"> • matchOffset and matchLength are only applicable to the following five constants: <ul style="list-style-type: none"> • DHCP_CLASS_HARDWARE • DHCP_CLASS_CLIENT_ID • DHCP_CLASS_VENDOR_ID • DHCP_CLASS_AGENT_CIRCUIT_ID • DHCP_CLASS_AGENT_REMOTE_ID • matchOffset and matchLength must be specified together.

Update DHCP Match Classes

A DHCP Match class' name property can be updated using the generic `update()` method.

Parameter	Description
properties	A string containing options, in addition to the following element: <ul style="list-style-type: none"> • ignoreError—If true, the validation errors for the available match values violating the match conditions will be ignored.

For more information, refer to [Updating Objects](#) on page 50.

Delete DHCP Match Classes

DHCP Match Classes can be deleted using the generic `delete()` method.

For more information, refer to [Deleting Objects](#) on page 52.

Add DHCP Sub Classes

Adds DHCP match class values.

Output / Response

Returns the object ID of the new DHCP match class value.

API Call:

```
long addDHCPSubClass ( long matchClassId, String matchValue, String properties )
```

Parameter	Description
matchClassId	The object ID of the match class in which the DHCP match class value is being defined.
matchValue	The value of the DHCP match value to be matched with the match class. The length of the match value must be equal to the length, in bytes, specified in the match class.
properties	A string containing the following element: <ul style="list-style-type: none"> • description—a description of the match class.

Update DHCP Sub Classes

A DHCP Sub class' **matchValue** and **description** properties can be updated using the generic `update()` method.

For more information, refer to [Updating Objects](#) on page 50.

Delete DHCP Sub Classes

DHCP Sub Classes can be deleted using the generic `delete()` method.

For more information, refer to [Deleting Objects](#) on page 52.

DHCP Raw Options

The DHCPv4 and DHCPv6 Raw options allow you to add DHCPv4 and DHCPv6 deployment options that are not directly supported by Address Manager.

A raw option is passed to the DHCP services on the managed server exactly as you type it in the **rawData** parameter. Therefore, it is essential that you enter the data with the correct syntax. There is no error checking or data checking on the raw option. In the event of a syntax error in a DHCP Raw option, DHCP service will stop then rollback to the previous DHCP configuration, resulting in a service outage.

Raw options are not inherited between levels. They cannot be set at the configuration level or the IP block level because options set at these levels would not be inherited below. They can be set at the following levels:

- **For DHCPv4 raw options:**
 - Server
 - Server group
 - IPv4 network
 - IPv4 network template
 - IPv4 DHCP range
 - DHCP match class
 - DHCP match class value
 - MacAddress
 - MacPool
 - IPv4 DHCP reserved address
- **For DHCPv6 raw options:**
 - Server
 - Server group
 - IPv6 network
 - IPv6 DHCP reserved address

To add or update DHCP raw deployment options, refer to [Add Raw Deployment Option](#) on page 144 and [Update Raw Deployment Option](#) on page 145.

Shared Networks

Shared network declarations in DHCP are used to group together different logical subnets that share the same physical network.

➔ **Note:** Shared networks are known as superscopes in Windows.

➔ **Note:** For shared networks, each member networks of the shared network must have the same type of DHCP deployment role assigned to its network object.

For example, consider a network with 250 workstations on a physical network with the logical address ID of 192.168.6.0/24. You need to add 100 workstationsto this physical network, but the only available subnet ID is 192.168.12.0/24. If the subnets were contiguous (that is, 192.168.6.0/24 and 192.168.7.0/ 24), you could modify the subnet mask to create a single logical subnet to accommodate the additional computers (192.168.6.0/23). However, the two network IDs are not contiguous.

By configuring a shared network, you can group the two networks together. The benefit is that your DHCP server can allocate IP addresses from the common shared network to any host on either of the networks, without the need to isolate the networks to different router interfaces.

Tag groups and tags are the mechanism by which subnets are grouped into DHCP shared networks. To use shared networks, you need to associate a single tag group with a configuration. A configuration can have many associated tags, but only one tag that is associated for the purpose of forming shared networks.

Link a Shared Network Tag

Links an IPv4 network with a shared network tag.

Output / Response

None.

API Call:

```
void shareNetwork ( long networkId, long tagId )
```

Parameter	Description
networkId	The object ID of the IPv4 network that is being linked with a shared network tag. If <i>networkId</i> is not valid, an error will be returned.
tagId	The object ID of the tag that is linked. If <i>tagId</i> is not valid, an error will be returned.

Unlink a Shared Network Tag

Unlinks the shared network tag from an IPv4 network.

Output / Response

None.

API Call:

```
void unshareNetwork ( long networkId )
```

Parameter	Description
networkId	The object ID of the IPv4 network that is being unlinked from a shared network tag. If <i>networkId</i> is not valid, an error will be returned.

Get Shared Networks

Returns multiple IPv4 networks linked to the given shared network tag.

Output / Response

Returns an array of entities of all the IPv4 networks linked to the given shared network tag. If no networks are found, an empty array will be returned.

API Call:

```
APIEntity[] getSharedNetworks ( long tagId )
```

Parameter	Description
tagId	The object ID of the tag that is linked with shared IPv4 networks. If <i>tagId</i> is not valid, an error will be returned.

DNS

The DNS service part of the Address Manager API implements DNS structures through views and zones. All supported DNS resource record types can be manipulated through this part of the API. Control of DNS options allows you to customize the DNS service.

DNS Views

Address Manager treats all DNS structures as views. Address Manager creates the default View clause in the **named.conf** files and then creates additional Views as needed to add customized DNS response groups to the structure.

Each view is a child of a configuration object. Beneath the views are zones, which can contain sub-zones or resource records. All views can be associated with an Access Control List (ACL) to filter client requests and provide different sets of DNS information in response to requests from different clients. BIND views and Address Manager views are the same. Views in Address Manager use the same IP address to deliver different DNS services to different clients depending on their IP address. In a Address Manager configuration, you can add views for each client group that requires filtering against an ACL. Address Manager differs from standard implementations of views in the area of zone transfers. Address Manager only assigns a single IP address for DNS to a managed server. To send notifications to a slave server to request a zone transfer, DNS/DHCP Servers recognize each other through the use of TSIG keys. The TSIG keys enable each DNS/DHCP Server appliance to be dealt with on an individual basis with high security in terms of data integrity and authentication. Address Manager handles all of the implementation details for these advanced features while providing views-based service on a single port and on a single address.

Add DNS View

Adds DNS Views.

Output / Response

Returns the object ID for the new DNS view.

API Call:

```
long addView ( long configurationId, String name, String properties )
```

Parameter	Description
configurationId	The object ID of the parent configuration in which this DNS view is located.
name	The name of the view.
properties	Adds object properties, including user-defined fields.

Update DNS View

A DNS view's **name** property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

DNS View Generic Methods

DNS views use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Updating Objects](#) on page 50.

Add Access Control List (ACL)


Adds an Access Control List to a view.

Output / Response

Returns the object ID for the new ACL object.

API Call:

```
long addACL( long configurationId, String name, String properties )
```

Parameter	Description
configurationId	The object ID of the configuration on which ACL need to be added.
name	The name of the ACL.
properties	<p>A string containing the comma-separated list of options in the following format:</p> <pre>aclValues=IP4Address, IP6Address, IP4Network's CIDR, IP6Network's CIDR, ACL's name, TSIG key's name, Predefined BIND ACL values</pre> <p>where:</p> <ul style="list-style-type: none"> • IP4Address—the IPv4 address. • IP6Address—the IPv6 address. • IP4Network's CIDR—the CIDR notation defining the IPv4 network. • IP6Network's CIDR—the CIDR notation defining the IPv6 network. • ACL's name—the name of the ACL (Example: <code>acl aclName</code>). • TSIG key's name—the name of the TSIG key (Example: <code>key TSIGName</code>). • Predefined BIND ACL values—Example: <code>'none', 'any', 'localhost', 'localnets' or 'All', 'None', 'Local Host', 'Local Networks'</code>. <p> Note: Use an exclamation mark (!) to exclude a certain option. For example, <code>!none, !acl aclName, !10/ 24</code>, etc.</p>

Update Access Control List (ACL)

An Access Control List (ACL) can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

DNS Zones

In Address Manager, DNS zones are child objects of views. Zones can contain sub-zones and resource records. Sub-zones can be created several layers deep, as required.

Add Entity for DNS Zones

Adds DNS zones.

`addEntity()` is a generic method for adding configurations, DNS zones, and DNS resource records. When using `addEntity()` to add a zone, you must specify a single zone name without any . (dot) characters. The parent object must be either a DNS view or another DNS zone.

Output / Response

Returns the object ID for the new DNS zone.

API call:

```
long addEntity( long parentId, APIEntity entity )
```

Parameter	Description
parentId	The object ID of the parent DNS view or DNS zone to which the zone is added.
entity	The zone name, without any . (dot) characters, to be added.

Add Zone

Adds DNS zones. When using `addZone()`, you can use . (dot) characters to create the top level domain and subzones.

Output / Response

Returns the object ID for the new DNS zone.

API Call:

```
long addZone( long parentId, String absoluteName, String properties )
```

Parameter	Description
parentId	The object ID for the parent object to which the zone is being added. For top-level domains, the parent object is a DNS view. For sub zones, the parent object is a top-level domain or DNS zone.
absoluteName	The complete FQDN for the zone with no trailing dot (for example, example.com).
properties	<p>Adds object properties, including a flag for deployment, an optional network template association, and user-defined fields in the format:</p> <pre>deployable=<true false> template=<template id> <userField>=<userFieldValue></pre> <p>The deployable flag is false by default and is optional. To make the zone deployable, set the deployable flag to true.</p>

Get Zones by Hint

Returns an array of accessible zones of child objects for a given `containerId` value.

Output / Response

Returns an array of zones based on the input argument without their properties fields populated, or returns an empty array if `containerId` is invalid. If no access right option is specified, the View access level will be used by default.

API Call:

```
APIEntity[] getZonesByHint( long containerId, int start, int count, String options )
```

Parameter	Description
containerId	The object ID for the container object. It can be the object ID of any object in the parent object hierarchy. The highest parent object can be the configuration level.
start	Indicates where in the list of objects to start returning objects. The list begins at an index of 0.
count	Indicates the maximum number of child objects that this method will return. The maximum number of child objects cannot exceed more than 10.
options	<p>A string containing options. The Option names available in the <code>ObjectProperties</code> are <code>ObjectProperties.hint</code>, <code>ObjectProperties.accessRight</code>, and <code>ObjectProperties.overrideType</code>. Multiple options can be separated by a (pipe) character. For example:</p> <pre>hint=ab overrideType=HostRecord accessRight=ADD</pre> <p>The values for <code>ObjectProperties.hint</code> option can be the prefix of a zone name. For example:</p> <pre>String options = ObjectProperties.hint + "=abc "</pre> <p>The values for the <code>ObjectProperties.accessRight</code> and <code>ObjectProperties.overrideType</code> options must be one of the constants listed in Access Right Values on page 189 and Object Types on page 209. For example:</p> <pre>String options = ObjectProperties.accessRight + "=" + AccessRightValues.AddAccess + " " + ObjectProperties.overrideType + "=" + ObjectTypes.HostRecord;</pre>

Update Zone

A DNS zone's **name** property can be updated using the generic `update()` method.

For more information, refer to [Updating Objects](#) on page 50.

Zone Generic Methods

DNS zones use the generic `get()` and `delete()` methods for entities.

For more information, refer to [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Get Key Signing Key

Returns a string containing all active Key Signing Keys (KSK) for a given `entityId` value in a specified output format with its start time and expire time, divided by a delimiter (|). The list of returned KSKs is sorted in descending order by expiry date.

Output / Response

Returns a string containing up-to two active KSK(s) of an entity in the following format:

```
KSK in specified format|create time|expire time.
```

API Call:

```
String[] getKSK( long entityId, String format )
```

Parameter	Description
entityId	The object ID of the entity associated with the KSK. The only supported entity types are Zone, IPv4 block, and IPv4 network.
format	The output format of the KSK of an entity. The value must be one of the constants listed in DNSSEC Key Format on page 199.

DNS Zone Templates

DNS zone templates provide a standard set of DNS records and options that can be maintained in a central location. These templates contain the same tabs as a regular zone except for the **Deployment Roles** tab.

Any settings that should be repeated in several zones, such as mail servers or web servers, should be added to a zone template so that the setting can be applied easily and consistently across zones. The records and options set in a zone template are created in any zones to which the template is applied.

If a record or an option is updated in the template, it is also updated in any zones to which the template applies. Modifying a template-applied record or option in a zone severs the link between the template and the modified record or option. If the template is re-applied, you have a choice to ignore conflicts or overwrite the conflicting objects with the settings in the template.

Zone templates can be edited to change the template name. Editing a zone template is exactly the same as building a zone, with the exception of assigning deployment roles, because zone templates are not deployable.

Add Zone Template

Adds a DNS zone template.

Output / Response

Returns the object ID of the new DNS zone template.

API Call:

```
long addZoneTemplate( long parentId, String name, String properties )
```

Parameter	Description
parentId	The object ID of the parent DNS view when adding a view-level zone template. The object ID of the configuration when adding a configuration-level zone template.
name	The name of the DNS zone template. This value can be an <i>empty string</i> ("").
properties	Adds object properties, including user-defined fields.

Assign or Update Template

Assigns, updates, or removes DNS zone and IPv4 network templates.

Output / Response

None.

API Call:

```
void assignOrUpdateTemplate( long entityId, long templateId, String properties )
```

Parameter	Description
entityId	The object ID of the IPv4 network to which the network template is to be assigned or updated, or the object ID of the zone to which the zone template is to be assigned or updated.
templateId	The object ID of the DNS zone template or IPv4 network template. To remove a template, set this value to 0 (zero).
properties	<p>A string containing the following settings:</p> <ul style="list-style-type: none"> • ObjectProperties.templateType—Specifies the type of template on which this operation is being performed. The possible values are <i>ObjectProperties.IP4NetworkTemplateType</i> (Assigning or updating IP4NetworkTemplate on an IP4Network) and <i>ObjectProperties.zoneTemplateType</i> (Assigning or updating zoneTemplate on a DNS zone). This is mandatory. <p>Along with ObjectProperties.templateType, user can also specify the reapply mode for various properties of the template.</p> <ul style="list-style-type: none"> • For Network template, the following additional parameters can also be specified: <ul style="list-style-type: none"> • <i>ObjectProperties.gatewayReapplyMode</i> • <i>ObjectProperties.reservedAddressesReapplyMode</i> • <i>ObjectProperties.dhcpRangesReapplyMode</i> • <i>ObjectProperties.ipGroupsReapplyMode</i> • <i>ObjectProperties.optionsReapplyMode</i> • For Zone Template, the following additional parameter can also be specified: <ul style="list-style-type: none"> • <i>ObjectProperties.zoneTemplateReapplyMode</i> <p>The possible values for re-apply mode properties are:</p> <ul style="list-style-type: none"> • <i>ObjectProperties.templateReapplyModeUpdate</i> • <i>ObjectProperties.templateReapplyModeIgnore</i> • <i>ObjectProperties.templateReapplyModeOverwrite</i> <p>If the re-apply mode is not specified in the properties, the default <i>ObjectProperties.templateReapplyModeIgnoremode</i> is used.</p> <p>➔ Note: If you are not using Java or Perl, refer to Object Properties on page 201 for the actual values.</p>

Java client example:

```
EntityProperties ntProp = new EntityProperties();
ntProp.addProperty( ObjectProperties.templateType,
ObjectProperties.IP4NetworkTemplateType );
ntProp.addProperty( ObjectProperties.gatewayReapplyMode,
ObjectProperties.templateReapplyModeUpdate );
ntProp.addProperty( ObjectProperties.reservedAddressesReapplyMode,
ObjectProperties.templateReapplyModeUpdate );
service.assignOrUpdateTemplate( ip4N20_26Id, networkTemplateId,
ntProp.getPropertiesString() );
```

Perl client example:

```
SOAP::Data->type( 'string' )->name( 'properties' )->
value( ObjectProperties::templateType."=".ObjectProperties::
IP4NetworkTemplateType."|".
ObjectProperties:: gatewayReapplyMode."=".ObjectProperties::
templateReapplyModeUpdate."|" )
->attr({xmlns => ''}) ->result;
```

Update Zone Template

A zone template's **name** property can be updated using the generic `update()` method.

For more information, refer to [Updating Objects](#) on page 50.

Zone Template Generic Methods

Zone templates use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Add Records to DNS Zone Template

`addEntity()` is a generic method for adding configuration, DNS zones, and DNS resource records.

You must use this method to add or update DNS resource records to DNS zone templates. For more information, refer to [Add Entity for Resource Records](#) on page 124.

ENUM Zones

ENUM zones provide voice over IP (VoIP) functionality within a DNS server.

The system requires DNS to manage the phone numbers associated with client end points; Address Manager provides an E164 or ENUM zone type for this purpose. The ENUM zone represents the area code for the phone prefixes and numbers stored in it. ENUM zones contain special sub-zones called prefixes that represent telephone exchanges and can contain the records for the actual devices.

VoIP devices are addressed in several ways. A uniform resource identifier (URI) string provides custom forward locator references for these devices as covered in RFC 3401. Reverse DNS is used to discover the relevant information for a device based on its phone number. Name authority pointer (NAPTR) records are used to represent this information.

Add ENUM Zone

Adds ENUM zones.

Output / Response

Returns the object ID for the new ENUM zone.

API Call:

```
long addEnumZone( long parentId, long prefix, String properties )
```

Parameter	Description
parentId	The object ID for the parent object of the ENUM zone.
prefix	The number prefix for the ENUM zone.
properties	Adds object properties, including user-defined fields.

Update ENUM Zone

An ENUM zone's **name** property can be updated using the generic `update()` method.

For more information, refer to [Updating Objects](#) on page 50.

ENUM Zone Generic Methods

ENUM zones use the generic `get()` and `delete()` methods for entities.

For more information, refer to [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

ENUM Numbers

ENUM number objects represent VoIP phone numbers within Address Manager. This functionality is provided as an alternative to using raw NAPTR records.

Add ENUM Number

adds ENUM numbers.

Output / Response

Returns the object ID for the new ENUM number record.

API Call:

```
long addEnumNumber( long parentId, int number, String properties )
```

Parameter	Description
parentId	The object ID of the parent object for the ENUM number. The parent object for an ENUM number is always an ENUM zone.
number	The ENUM phone number.
properties	Adds object properties, and user-defined fields, including the data string, which includes service , URI , comment and ttl values.

Update ENUM Number

You can update an ENUM number's **number** property using the generic `update()` method.

For more information, refer to [Updating Objects](#) on page 50.

ENUM Number Generic Methods

ENUM numbers use the generic `get()` and `delete()` methods for entities.

For more information, refer to [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

DNS Resource Records

In Address Manager, DNS resource records describe the characteristics of a zone or sub-zone. Address Manager supports the following types of resource records.

Resource Record Type	Description
Host Record	Designates an IP address for a device. A new host requires a name and an IP address. Multiple addresses may exist for the same device.
Mail Exchanger Record	Designates the host name and preference for a mail server or exchanger. An MX record requires a name and a priority value.

Resource Record Type	Description
	Priorities with a lower numeric value are chosen first in assessing delivery options.
CNAME Alias Record	Specifies an alias for a host name. The alias record type requires a name.
Service Record	Defines services available within a zone, such as LDAP. A service record requires a name, priority, port, and weight. A lower priority value indicates precedence. The port value indicates the port on which the service is available. The weight value is used when multiple services have the same priority value; a higher weight value indicates precedence.
HINFO Host Info Record	Specifies optional text information about a host. The host info record includes CPU and OS information.
TXT Text Record	Associates arbitrary text with a host name. A text record includes name and text information. This record is used to support record types such as those used in Sender Policy Framework (SPF) e-mail validation.
Generic Record	The following generic record types are available: A, A6, AAAA, AFSDDB, APL, CAA, CERT, DHCID, DNAME, DS, IPSECKEY, ISDN, KEY, KX, LOC, MB, MG, MINFO, MR, NS, NSAP, PTR, PX, RP, RT, SINK, SPF, SSHFP, TLSA, WKS, and X25. These records contain name, type, and value information.
NAPTR Naming Authority Pointer Record	Specifies settings for applications, such as VoIP. These records are used in Address Manager to populate ENUM zones.

Handling dotted resource records

The resource record adder includes two new property parameters, namely, `parentZoneName` and `linkedParentZoneName`. Both parameters should be absolute names. The record absolute name must end with the zone absolute name; otherwise an exception is thrown, and an error message appears. **Record name is not subset of parent zone: abc.bcn.example.com.**

- ➔ **Note:** The `linkedParentZoneName` parameter must be used only for adding or updating the linked records for CName, MX and SRV records.

The `parentZoneName` parameter refers to a destination zone for the record to be added to, and applies to all resource record you add. This parameter makes it possible to support dot-separated record names. For example, an API user can add a resource record named **abc.xyz.example.com** to the zone **example.com**.

- ➔ **Note:** You should not use `parentZoneName` when updating records, because you cannot change the parent zone of a record. If you try to do this an exception is thrown.

Example 1:

Suppose the zone **example.com** has a dot separated host record **abc.xyz**. A user add a sub-zone called **xyz**, and then adds a host record **abc** to this sub-zone. There are now two host records named **abc.xyz.example.com**.

If an API user wants to link a CName record to **abc.xyz.example.com**, the linked record will be the one located in the sub-zone, because the user cannot link the record to the one in the parent zone. To allow API users to choose whatever parent zone they want, use the `linkedParentZoneName` parameter.

- ➔ **Note:** Use this parameter with CName, MX and SRV records. It cannot be used as metadata fields for these records. If this parameter is used in updating other resource records, an error occurs.

Example 2:

An API entity CName **abcName.example.com** has the following property string:

```
ttl=123|absoluteName=abcName.example.com|
linkedRecordName=bcnhost.dot.bcn.com|
```

An API user wants to change the linked record name to **abc.bcn.example.com**. The user applies the following updates to the property string:

```
linkedParentZoneName=example.com|absoluteName=abcName.example.com|ttl=123|
linkedRecordName=abc.bcn.example.com|
```

If the API user does not use the linkedParentZoneName parameter, Address Manager chooses the internal host record or alias record if it exists; otherwise it chooses the external host record for the linked record.

Generic Resource Records

Each resource record type has a specialized add method, but there are two general methods for adding resource records in the Address Manager API.

- [Add Resource Records](#) on page 123
- [Add Entity for Resource Records](#) on page 124

Add Resource Records


This method is a generic method for adding resource records of any kind by specifying the name, type, and rdata arguments.

Output / Response

Returns the object ID for the new resource record.

API Call:

```
long addResourceRecord( long viewId, String absoluteName, String type, String rdata, long ttl,
String properties )
```

Parameter	Description
viewId	The object ID for the parent view to which the resource record is being added.
absoluteName	The absolute name of the record, specified as an FQDN. If you are adding a record in a zone that is linked to an incremental Naming Policy, a single hash (#) sign must be added at the appropriate location in the FQDN. Depending on the policy order value, the location of the single hash (#) sign varies.
type	<p>The type of record being added. Valid values for this parameter are the resource record types shown in Object Types on page 209:</p> <ul style="list-style-type: none"> • AliasRecord • HINFORecord • HostRecord • MXRecord • TXTRecord <p> Note: To add <i>NAPTRRecord</i>, <i>SRVRecord</i> and <i>GenericRecord</i>, you must use <code>addNAPTRRecord()</code>, <code>addSRVRecord()</code> and <code>addGenericRecord()</code> methods respectively.</p>
rdata	The data for the resource record in BIND format (for example, for A records, A 10.0.0.4). You can specify either a single IPv4 or IPv6 address for the record.

Parameter	Description
ttl	The time-to-live value for the record. To ignore the ttl, set this value to -1.
properties	Adds object properties, including user-defined fields.

Add Entity for Resource Records

`addEntity()` is a generic method for adding configurations, DNS zones, and DNS resource records. Use this method to add resource records that have . (dot) characters in their names.



Note: In order to add DNS resource records to a DNS zone template, you must use the `addEntity()` method.

Output / Response

Returns the object ID for the new resource record.

API Call:

```
long addEntity( long parentId, APIEntity entity )
```

Parameter	Description
parentId	The object ID of the parent zone to which the record is added.
APIEntity	The resource record object being passed to the database.

Move Resource Records

Moves resource records between different zones that already exist.

Output / Response

None.

API Call:

```
void moveResourceRecord( long resourceRecordId, String destinationZone )
```

Parameter	Description
resourceRecordId	The object ID of the resource record to be moved.
destinationZone	Fully qualified domain name of the destination DNS zone to which the resource record will be moved.

NAPTR Records

NAPTR records specify settings for applications, such as VoIP. Address Manager uses NAPTR records to populate ENUM zones.

Add NAPTR Record

Adds NAPTR records.

This method will add the record under a zone. In order to add records under templates, you must use [Add Entity for Resource Records](#) on page 124.

Output / Response

Returns the object ID for the new NAPTR resource record.

API Call:

```
long addNAPTRRecord( long viewId, String absoluteName, int order, int preference, String service, String regexp, String replacement, String flags, long ttl, String properties )
```

Parameter	Description
viewId	The object ID for the parent view to which this record is added.
absoluteName	The FQDN for the record. If you are adding a record in a zone that is linked to a incremental Naming Policy, a single hash (#) sign must be added at the appropriate location in the FQDN. Depending on the policy order value, the location of the single hash (#) sign varies.
order	Specifies the order in which NAPTR records are read if several are present and are possible matches. The lower ordervalue takes precedence.
preference	Specifies the order in which NAPTR records are read if the ordervalue are the same in multiple records. The lower preferencevalue takes precedence.
service	Specifies the service used for the NAPTR record. Valid settings for this parameter are listed in ENUM Services on page 200.
regexp	A regular expression, enclosed in double quotation marks, used to transform the client data. If a regular expression is not specified, a domain name must be specified in the replacement parameter.
replacement	Specifies a domain name as an alternative to the regexp. This parameter replaces client data with a domain name.
flags	An optional parameter used to set flag values for the record.
ttl	The time-to-live value for the record. To ignore the ttl, set this value to -1.
properties	Adds object properties, including comments and user-defined fields.

Update NAPTR Record

A NAPTR record's **name**, **ttl**, **comment**, **order**, **preference**, **service**, **regexp**, and **replacement** properties can be updated using the generic `update()` method.

For more information, refer to [Updating Objects](#) on page 50.

NAPTR Record Generic Methods

NAPTR records use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

External Host Records

External hosts, specified with an FQDN, represent host names that reside outside the Address Manager-managed servers.

External hosts are never deployed, but act as glue that lets other records (such as MX and CNAME records) link to hosts that are not managed by Address Manager. Any references within a view that refer to entities completely outside of the IP space and DNS namespace managed by Address Manager are defined here.

Add External Host Record

Adds external host records.

This method will add the external host record under a zone. In order to add external host records under templates, you must use [Add Entity for Resource Records](#) on page 124.

Output / Response

Returns the object ID for the new external host record.

API Call:

```
long addExternalHostRecord( long viewId, String name, String properties )
```

Parameter	Description
viewId	The object ID for the parent view to which this record is being added.
name	The FQDN for the host record.
properties	Adds object properties, including comments and user-defined fields.

Get External Host Records associated with IP addresses

Returns an array of external host records that are associated with IPv4 or IPv6 addresses.

You can use the `getLinkedEntities()` method to retrieve the array of external host records that are assigned to IP addresses. Use **ObjectTypes.ExternalHostRecord** to retrieve all linked external host records with IP addresses. For example:

```
APIEntity[] linkedEntities = service.getLinkedEntities( ipAddressId,
    ObjectTypes.ExternalHostRecord, 0, Integer.MAX_VALUE );
```

Get IP address assigned with External Host Records

Retrieves IP addresses that are assigned with external host records.

Use the following methods to get the address linked with external host records:

- `getEntityById()`
- `getEntityByName()`
- `getEntities()`
- `getEntitiesByName()`

Update External Host Record

An external host record's **name** and **comment** properties can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

Update External Host Records assigned to IP addresses

You can update external host records that are assigned to IPv4 or IPv6 addresses using the generic `update()` method. Refer to the following *ptrs* property values and example:

- **null**—updates IPv4 or IPv6 address without any change.
- **empty string ("")**—use an empty string (") for the *ptrs* property value when updating. This will remove all linked external host records.
- **Valid comma-separated external host records string**—use a valid string. This will remove previously linked external host records and link newly specified external host records. The records previously linked should not be unlinked.

```
EntityProperties props = new EntityProperties();
props.addProperty( ObjectProperties.ptrs,
    "123,exHostFQDN.com,456,exHostFQDN.net" );
service.update( ipAddress );
```

External Host Record Generic Methods

External host records use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Host Records

A host record, or A record, designates an IP address for a device.

A new host requires a name and an IP address. Multiple addresses may exist for the same device. Set the time-to-live for this record to an override value here so that the record has a longer or shorter ttl. A comment field is also included.

Add Host Record

Adds host records for IPv4 or IPv6 addresses. All addresses must be valid addresses.

This method will add the record under a zone. In order to add records under templates, you must use [Add Entity for Resource Records](#) on page 124.

When adding a host record, the **reverseRecord** property, if not explicitly set in the **properties** string, is set to **true** and Address Manager creates a reverse record automatically. IPv4 addresses can be added in both workflow and non-workflow mode. IPv6 addresses can be added in non-workflow mode only. For more information on workflow mode, see [Workflow Change Requests](#) on page 182.

Output / Response

Returns the object ID for the new host resource record.

API call:

```
long addHostRecord ( long viewId, String absoluteName, String addresses, long ttl, String properties)
```

Parameter	Description
viewId	The object ID for the parent view to which this record is being added.
absoluteName	The FQDN for the host record. If you are adding a record in a zone that is linked to a incremental Naming Policy, a single hash (#) sign must be added at the appropriate location in the FQDN. Depending on the policy order value, the location of the single hash (#) sign varies.
addresses	A list of comma-separated IP addresses (for example, 10.0.0.5,130.4.5.2).
ttl	The time-to-live value for the record. To ignore the ttl, set this value to -1.
properties	Adds object properties, including comments and user-defined fields.

Add Bulk Host Records

Adds host records using auto-increment from the specific starting address.

This method will add the record under a zone. In order to add records under templates, you must use [Add Entity for Resource Records](#) on page 124.

This method adds host records to a zone linked to a DNS naming policy, each with an IP address auto-incremented starting from a specific address in a network.

Output / Response

Returns an array of host record APIEntity objects based on available addresses and number of IP addresses required. If no addresses are available, an error will be shown.

API call:

```
APIEntity[] addBulkHostRecord ( long viewId, String absoluteName, long ttl, long networkId, String startAddress, int numberOfAddresses, String properties)
```

Parameter	Description
viewId	The object ID for the parent view to which this record is being added.
absoluteName	The FQDN for the host record. If you are adding a record in a zone that is linked to a incremental Naming Policy, a single hash (#) sign must be added at the appropriate location in the FQDN. Depending on the policy order value, the location of the single hash (#) sign varies.
ttl	The time-to-live value for the record. To ignore the ttl, set this value to -1 .
networkId	The network which to get the available IP addresses. Each address is used for one host record.
startAddress	The starting IPv4 address for getting the available addresses.
numberOfAddresses	The number of addresses.
properties	excludeDHCPRange=true/false , if true then IP addresses within a DHCP range will be skipped. This argument can also contain user-defined fields.

Get Host Record by Hint

Returns an array of objects with host record type.

Output / Response

Returns an array of host record APIEntity objects.

API call:

```
APIEntity[] getHostRecordsByHint ( int start, int count, String options)
```

Parameter	Description
start	Indicates where in the list of objects to start returning objects. The list begins at an index of 0.
count	Indicates the maximum of child objects that this method will return. The value must be less than or equal to 10.
options	<p>A string containing options. The supported options are hint and retrieveFields. Multiple options can be separated by a (pipe) character. For example:</p> <pre>hint=^abc retrieveFields=false</pre> <p>If the hint option is not specified in the string, searching criteria will be based on the same as zone host record. The following wildcards are supported in the hint option.</p> <ul style="list-style-type: none"> • ^—matches the beginning of a string. For example: ^ex matches example but not text. • \$—matches the end of a string. For example: ple\$ matches example but not please. • ^ \$—matches the exact characters between the two wildcards. For example: ^example\$ only matches example. • ?—matches any one character. For example: ex?t matches exit. • *—matches one or more characters within a string. For example: ex*t matches exit and excellent. <p>The default value for the retrieveFields option is set to <i>false</i>. If the option is set to <i>true</i>, user-defined field will be returned. If the options string does not contain retrieveFields, user-defined field will not be returned.</p>

Get IP Address with Host Records

Returns an array of IP addresses with linked records and the IP addresses that are assigned as DHCP Reserved, Static or Gateway.

Output / Response

Returns an array of IP address APIEntity objects with their linked host records and the IP addresses that are assigned as DHCP Reserved, Static or Gateway. The output has the following format:

hostId:hostName:zoneId:zoneName:viewId:viewName:hasAlias;

API call:

```
APIEntity[] getNetworkLinkedProperties ( long networkId )
```

Parameter	Description
networkId	The object ID for the IPv4 network.

Get Dependent Records

Returns any CNAME, MX, or SRV resource records that reference the specified host record.

- ➔ **Note:** This method is deprecated. Using this method now returns an error message. Use the `getLinkedEntities()` method instead. For more information, see [Get Linked Entities](#) on page 53.

Output / Response

Returns an array of APIEntity objects representing the CNAME, MS, or SRV records referencing the specified host record.

API call:

```
APIEntity[] getDependentRecords ( long entityId, int start, int count )
```

Parameter	Description
entityId	The object ID for the host record for which you want to retrieve dependent records.
start	Indicates where in the list of dependent records to start returning objects. This list begins at an index of 0.
count	The maximum number of dependent records to return.

Update Host Record

A host records's **name**, **ttl**, **comment**, and **addresses** properties can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

Host Record Generic Methods

Host records use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Alias Records

This is a Canonical Name or CNAME record, used to specify an alias for a host name.

The alias record type requires only a name to be supplied. The time-to-live for this record can be set to an override value so that this record has a longer or shorter ttl. A comment field is also included.

Add Alias Record

Adds alias records.

This method attempts to link to an existing host record. If an existing host record cannot be located, the method attempts to link to an existing external host record. If neither can be located, the method fails. This method will add the record under a zone. In order to add records under templates, you must use [Add Entity for Resource Records](#) on page 124.

Output / Response

Returns the object ID for the new alias resource record.

API call:

```
long addAliasRecord ( long viewId, String absoluteName, String linkedRecordName, long ttl,
String properties )
```

Parameter	Description
viewId	The object ID for the parent view to which this record is being added.
absoluteName	The FQDN of the alias. If you are adding a record in a zone that is linked to a incremental Naming Policy, a single hash (#) sign must be added at the appropriate location in the FQDN. Depending on the policy order value, the location of the single hash (#) sign varies.
linkedRecordName	The name of the record to which this alias will link.
ttl	The time-to-live value for the record. To ignore the ttl, set this value to -1 .
properties	Adds object properties, including comments and user-defined fields.

Get Aliases by Hint

Returns an array of CNAMEs with linked record name.

Output / Response

Returns an array of Alias APIEntity objects.

API call:

```
APIEntity[] getAliasesByHint ( int start, int count, String options )
```

Parameter	Description
start	indicates where in the list of objects to start returning objects. The list begins at an index of 0.
count	indicates the maximum of child objects that this method will return. The value must be less than or equal to 10.
options	<p>a string containing options. The supported options are hint and retrieveFields. Multiple options can be separated by a (pipe) character. For example:</p> <pre>hint=^abc retrieveFields=false</pre> <p>If the hint option is not specified in the string, searching criteria will be based on the same as zone alias. The following wildcards are supported in the <i>hint</i> option.</p> <ul style="list-style-type: none"> ^—matches the beginning of a string. For example: ^ex matches example but not text. \$—matches the end of a string. For example: ple\$ matches example but not please.

Parameter	Description
	<ul style="list-style-type: none"> • ^ \$—matches the exact characters between the two wildcards. For example: ^example\$ only matches example. • ?—matches any one character. For example: ex?t matches exit. • *—matches one or more characters within a string. For example: ex*t matches exit and excellent. <p>The default value for the retrieveFields option is set to <i>false</i>. If the option is set to <i>true</i>, user-defined field will be returned. If the options string does not contain retrieveFields, user-defined field will not be returned.</p>

Update Alias Record

An alias record's **name**, **ttl**, **comment**, and **linked record** properties can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

Alias Record Generic Methods

Alias records use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Text Records

Text records associate arbitrary text with a host name.

This record features **name** and **text** fields and is used to support record types such as those used in SPF email validation. The time-to-live for this record can be set to an override value here so that this record has a longer or shorter ttl. A comment field is also included.

Add Text Record

Adds TXT records.

This method will add the record under a zone. To add records under templates, you must use [Add Entity for Resource Records](#) on page 124.

Output / Response

Returns the object ID for the new TXT record.

API call:

```
long addTXTRecord ( long viewId, String absoluteName, String txt, long ttl, String properties )
```

Parameter	Description
viewId	The object ID for the parent view to which the record is being added.
absoluteName	The FQDN of the text record. If you are adding a record in a zone that is linked to a incremental Naming Policy, a single hash (#) sign must be added at the appropriate location in the FQDN. Depending on the policy order value, the location of the single hash (#) sign varies.
txt	The text data for the record.
ttl	The time-to-live value for the record. To ignore the ttl, set this value to -1.
properties	Adds object properties, including comments and user-defined fields.

Update Text Record

A text record's **name**, **ttl**, **comment**, and **text data** properties can be updated using the generic `update()` method. For more information, see [Updating Objects](#) on page 52.

For more information, see [Updating Objects](#) on page 50.

Text Record Generic Methods

Text records use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

HINFO Records

The Host Info or HINFO resource record contains optional text information about a host.

The standard version of this record has **name**, **cpu** and **os** fields. The time-to-live for this record can be set to an override value here so that it has a longer or shorter ttl. A comment field is also included. This record type is a good candidate for user-defined fields to track information about hosts on the network.

Add HINFO Record

Adds HINFO records.

Output / Response

Returns the object ID for the new HINFO resource record.

API call:

```
long addHINFORecord ( long viewId, String absoluteName, String cpu, String os, long ttl, String properties )
```

Parameter	Description
viewId	The object ID for the parent view to which the HINFO record is being added.
absoluteName	The FQDN of the HINFO record. If you are adding a record in a zone that is linked to a incremental Naming Policy, a single hash (#) sign must be added at the appropriate location in the FQDN. Depending on the policy order value, the location of the single hash (#) sign varies.
cpu	A string providing central processing unit information.
os	A string providing operating system information.
ttl	The time-to-live value for the record. To ignore the ttl, set this value to -1.
properties	Adds object properties, including comments and user-defined fields.

Update HINFO Record

An HINFO record's **name**, **ttl**, **comment**, **cpu**, and **os** properties can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

HINFO Record Generic Methods

HINFO records use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

MX Records

A Mail Exchanger or MX record designates the host name and preference value for a mail server or exchanger, as defined in RFC 974.

An MX Record requires a name and a priority value. Priorities with a lower numeric value are chosen first in assessing delivery options. The time-to-live for this record can be set to an override value, so that this record has a longer or shorter ttl. A comment field is also included. This method attempts to link to an existing host record. If an existing host record cannot be located, the method attempts to link to an existing external host record. If neither can be located, the method fails.

Add MX Record

Adds MX records.

This method will add the record under a zone. In order to add records under templates, you must use [Add Entity for Resource Records](#) on page 124.

Output / Response

Returns the object ID for the new MX resource record.

API call:

```
long addMXRecord ( long viewId, String absoluteName, int priority, String linkedRecordName, long ttl, String properties )
```

Parameter	Description
viewId	The object ID for the parent view to which the MX record is being added.
absoluteName	The FQDN for the record. If you are adding a record in a zone that is linked to a incremental Naming Policy, a single hash (#) sign must be added at the appropriate location in the FQDN. Depending on the policy order value, the location of the single hash (#) sign varies.
priority	Specifies which mail server to send clients to first when multiple matching MX records are present. Multiple MX records with equal priority values are referred to in a round-robin fashion.
linkedRecordName	The FQDN of the host record to which this MX record is linked.
ttl	The time-to-live value for the record. To ignore the ttl, set this value to -1 .
properties	Adds object properties, including comments and user-defined fields.

Update MX Record

An MX record's **name**, **ttl**, **comment**, **linked record**, and **priority** properties can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

MX Record Generic Methods

MX records use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

SRV Records

Service records define services that are available within the zone, such as LDAP.

An SRV record requires a name by which it is known in Address Manager. The time-to-live for this record can be set to an override value here so that this record has a longer or shorter ttl. A comment field is

also included. This method attempts to link to an existing host record. If an existing host record cannot be located, the method attempts to link to an existing external host record. If neither can be located, the method fails.

Add SRV Record

Adds SRV records.

This method will add the record under a zone. In order to add records under templates, you must use [Add Entity for Resource Records](#) on page 124.

Output / Response

Returns the object ID for the new SRV record.

API call:

```
long addSRVRecord ( long viewId, String absoluteName, int priority, int port, int weight, String linkedRecordName, long ttl, String properties )
```

Parameter	Description
viewId	The object ID for the parent view to which the SRV record is being added.
absoluteName	The FQDN of the SRV record. If you are adding a record in a zone that is linked to a incremental Naming Policy, a single hash (#) sign must be added at the appropriate location in the FQDN. Depending on the policy order value, the location of the single hash (#) sign varies.
priority	Specifies which SRV record to use when multiple matching SRV records are present. The record with the lowest value takes precedence.
port	The TCP/UDP port on which the service is available.
weight	If two matching SRV records within a zone have equal priority, the weight value is checked. If the weight value for one object is higher than the other, the record with the highest weight has its resource records returned first.
linkedRecordName	The FQDN of the host record to which this SRV record is linked.
ttl	The time-to-live value for the record. To ignore the ttl, set this value to -1.
properties	Adds object properties, including comments and user-defined fields.

Update SRV Record

An SRV record's **name**, **ttl**, **comment**, **linked record**, **priority**, **port**, and **weight** properties can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

SRV Record Generic Methods

SRV records use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Start of Authority Records

Start of Authority (SOA) records are used to define administrative information relating to particular zones.

Add Start of Authority Record

Adds SOA records.

Output / Response

Returns the object ID for the new SOA record.

API call:

```
long addStartOfAuthority ( long parentId, String email, long refresh, long retry, long expire, long minimum, String properties )
```

Parameter	Description
parentId	The object ID of the parent object of the SOA record.
email	Specifies the email address of the administrator for the zones to which the SOA applies.
refresh	The amount of time that a slave server waits before attempting to refresh zone files from the master server. This is specified in seconds using a 32-bit integer value. RFC 1912 recommends a value between 1200 and 4300 seconds.
retry	Specifies the amount of time that the slave server should wait before re-attempting a zone transfer from the master server after the refresh value has expired. This is specified as a number of seconds using a 32-bit integer value.
expire	Specifies the length of time that a slave server will use a non-updated set of zone data before it stops sending queries. This is specified as a number of seconds using a 32-bit integer. RFC 1912 recommends a value from 1209600 to 2419200 seconds or 2 to 4 weeks.
minimum	Specifies the maximum amount of time that a negative cache response is held in cache. A negative cache response is a response to a DNS query that does not return an IP address (a failed request). Until this value expires, queries for this DNS record return an error. The maximum value for this field is 10800 seconds, or 3 hours.
properties	Adds object properties, including comments and user-defined fields. The supported properties are <i>time-to-live (TTL)</i> , <i>primary server (mname)</i> and <i>serial number format (serialNumberFormat)</i> . To override the default TTL value for SOA records, use <i>ObjectProperties.ttl=<value></i> in the properties string.

Update Start of Authority Record

Use the generic `update()` method to update an SOA record.

The following properties of an SOA record can be updated using the generic `update()` method:

- **name**
- **email**
- **refresh**
- **retry**
- **expire**
- **minimum**
- **mname**
- **serialNumberFormat**

For more information, see [Updating Objects](#) on page 50.

Start of Authority Record Generic Methods

SOA records use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Generic Records

Use the generic resource record methods to add and update the following resource record types: A6, AAAA, AFSDDB, APL, CAA, CERT, DNAME, DNSKEY, DS, ISDN, KEY, KX, LOC, MB, MG, MINFO, MR, NS, NSAP, PX, RP, RT, SINK, SSHFP, TLSA, WKS, and X25.

The fields available are **name**, **type** (which defines the custom record type), and **data** (the rdata value for the custom type). The time-to-live for this record can be set to an override value, so the record has a longer or shorter ttl. A comment field is also included.

Add Generic Record

Adds Generic records.

Output / Response

Returns the object ID for the new generic resource record.

API call:

```
long addGenericRecord (long viewId, String absoluteName, String type, String rdata, long ttl, String properties)
```

Parameter	Description
viewId	The object ID for the parent view to which the record is being added.
absoluteName	The FQDN of the record. If you are adding a record in a zone that is linked to a incremental Naming Policy, a single hash (#) sign must be added at the appropriate location in the FQDN. Depending on the policy order value, the location of the single hash (#) sign varies.
type	The type of record being added. Valid settings for this parameter are the generic resource record types supported in Address Manager: A6, AAAA, AFSDDB, APL, CAA, CERT, DNAME, DNSKEY, DS, ISDN, KEY, KX, LOC, MB, MG, MINFO, MR, NS, NSAP, PX, RP, RT, SINK, SSHFP, TLSA, WKS, and X25.
rdata	The data for the resource record in BIND format (for example, for A records, 10.0.0.4).
ttl	The time-to-live value for the record. To ignore the ttl, set this value to -1.
properties	Adds object properties, including comments and user-defined fields.

Update Generic Record

A generic record's **name**, **type**, **rdata**, **ttl**, and **comment** properties can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

Generic Record Generic Methods

Generic records use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

DNS Options

DNS options define the deployment of Address Manager DNS services. Address Manager supports most of the options used by both BIND and Microsoft DNS.

For options that are not directly supported by Address Manager, you can use the raw deployment option. For more information, refer to [Add Raw Deployment Option](#) on page 144 and [Update Raw Deployment Option](#) on page 145.

Add DNS Option


Adds DNS options.

Output / Response

Returns long type and represents the database object ID of the newly added DNS deployment option.

API Call:

```
long addDNSDeploymentOption( long entityId, String name, String value, String properties )
```

Parameter	Description
entityId	The object ID for the entity to which this deployment option is being added.
name	The name of the DNS option being added. This name must be one of the constants listed in DNS Options on page 197.
value	The value being assigned to the option.  Note: Depending on the type of DNS deployment option being added, the format of the value input might differ. For more information, refer to Reference: DNS Option value formats on page 138.
properties	Adds object properties, including comments and user-defined fields.

Get DNS Option


Retrieves all DNS options assigned for the object specified excluding the options inherited from the higher-level parent object.

Output / Response

Returns an instance of the type **APIDeploymentOption** that represents the DNS deployment option or empty if none were found.

API Call:

```
APIDeploymentOption getDNSDeploymentOption( long entityId, String name, long serverId )
```

Parameter	Description
entityId	The object ID for the entity to which this deployment option is assigned.
name	The name of the DNS option. This name must be one of the constants listed in DNS Options on page 197.  Note: Depending on the type of DNS deployment option being retrieved, the format of the value might differ. For more information, refer to Reference: DNS Option value formats on page 138.
serverId	Specifies the server or server group to which this option is assigned. To retrieve an option that has not been assigned to a server role, set this value to 0 (zero). Omitting this parameter from the method call will result in an error.

Update DNS Option

Updates DNS options.

Output / Response

None.

API Call:

```
void updateDNSDeploymentOption( APIDeploymentOption option )
```

Parameter	Description
option	The object ID of the DNS option to be updated.

Delete DNS Option

Deletes DNS options.

Output / Response

None.

API Call:

```
void deleteDNSDeploymentOption( long entityId, String name, long serverId )
```

Parameter	Description
entityId	The object ID for the entity to which the deployment option is assigned.
name	The name of the DNS option being deleted. This name must be one of the constants listed in DNS Options on page 197.
serverId	Specifies the server or server group to which the option is assigned. To delete an option that has not been assigned to a server role, set this value to 0 (zero). Omitting this parameter from the method call will result in an error.

Reference: DNS Option value formats

The input and output value formats for DNS Option API methods.

When performing an add operation of DNS deployment options, the value input is in double quotation marks ("). For example, adding a Lame TTL DNS deployment option with a value of "300" using the `addDNSDeploymentOption` API method to Address Manager would look similar to the following:

Input

```
long optId = service.addDNSDeploymentOption(100977, DNSOptions.LAME_TTL,
    "300", "");
```

Similarly, retrieving a Lame TTL DNS deployment option with a value of "300" using the `getDeploymentOptions` API method would return values similar to the following:

Output

```
{
  "id": 100977,
  "type": "DNS",
  "name": "lame-ttl",
  "value": "300",
  "properties": "inherited=false|"
```

```
}
```

Exception - When performing an add, update, or get API call with the Root Hints (CACHE) DNS deployment option, the value input is in double quotation marks and braces ("{}"). For example, adding a Root Hint DNS deployment option with a specified name value of "admin.corp" and IP address of 172.25.19.53 using the `addDNSDeploymentOption` API method would look similar to the following:

Input

```
logn optId = service.addDNSDeploymentOption(100977, DNSOptions.CACHE,
    "{admin.corp,172.25.19.53}", "");
```

Similarly, retrieving a Root Hint DNS deployment option with a specified name value of "admin.corp" and IP address of 172.25.19.53 using the `getDeploymentOption` API method would return values similar to the following:

Output

```
{
  "id": 100977,
  "type": "DNS",
  "name": "cache",
  "value": "{admin.corp,172.25.19.53}",
  "properties": "inherited=false|"
}
```



Note: When adding a Root Hint DNS deployment option with the value of "Auto", the value defined in the add API method must be empty double quotation marks and braces ("{}"). Similarly, when performing a get API method of DNS deployment options where the Root Hint has a value of "Auto", the value returned is empty double quotation marks and braces ("{}").

DNS Raw Option

The DNS raw option allows you to add options to the DNS service in a raw format that gets passed to the service when deployed. Address Manager does not perform any data checking on raw options. You must ensure that the syntax for these options is correct.

Raw options assigned to a parent object are not inherited by child objects. For example, an option set at a parent zone is not inherited by a child zone. However, raw options assigned to a server group are inherited *only* by the servers that are linked to that server group.

DNS raw options can be set at the following levels:

- Server
- Server group
- DNS View
- DNS zone
- Root zone
- ENUM zone
- Zone template
- IPv4 block
- IPv4 network
- IPv4 network template
- IPv6 block
- IPv6 network



Note: DNS raw options cannot be configured for Windows servers.

To add or update DNS raw deployment options, refer to [Add Raw Deployment Option](#) on page 144 and [Update Raw Deployment Option](#) on page 145.

DNS Response Policies

The Response Policies feature allows users to manage a recursive DNS resolver attempting to respond to the queries that might not be desirable or legal.

You can set the types of response policies based on your needs and deploy to a DNS server managed under Address Manager. By setting up these response policies, you can block, redirect, or allow particular domain name queries that you wish to and must prevent. For example:

- If you are a corporate user and want to prevent employees from being connected to any harmful website, you can setup the response policies and block these harmful websites so that they does not return the query response or the employees can simply be redirected to an appropriate website.
- If you need to follow a government regulation that mandates certain DNS blocking, the response policies can be used to implement this requirement.

There are three different types of response policies that can be set based on user requirements:

Blacklist

Matching items in the list of blacklist object return an NXDomain result.

Blackhole

Matching items in this response policy object return a NOERROR result with no answers.

Whitelist

Matching items in this response policy object are excluded from further processing.

Add Response Policy


Adds a DNS response policy.

Output / Response

Returns the object ID of the new DNS response policy added.

API Call:

```
long addResponsePolicy( long configurationId, String name, String responsePolicyType,
long ttl, String properties )
```

Parameter	Description
configurationId	The object ID of the configuration to which the response policy is being added.
name	The name of the DNS response policy being added.
responsePolicyType	The type of response policy being added. The available values are BLACKLIST, BLACKHOLE and WHITELIST.  Note: The responsePolicyType values need to be in <i>CAPITAL</i> letters.
ttl	The time-to-live value in seconds.
properties	A string containing options, including comments and user-defined fields.

Update Response Policy

A response policy's **name**, **ttl** and **responsePolicyType** properties can be updated using the generic `update()` method.

For more information, refer to [Updating Objects](#) on page 50.

Response Policy Generic Methods

Response policy uses the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Upload Response Policy Item


Uploads one response policy file containing a list of fully qualified domain names (FQDNs).

Output / Response

None.

API Call:

```
void uploadResponsePolicyItems( long parentId, byte[] policyItemsData )
```

Parameter	Description
parentId	The object ID of the parent response policy under which the response policy item file is being uploaded.
policyItemsData	<p>The file to be uploaded under the response policy. This file is passed to Address Manager as a byte array.</p> <p> Note: This actual file size should be no more than 75 MB.</p>

Search Response Policies

Searches Response Policy items configured in local Response Policies or predefined BlueCat Security feed data. The search will return a list of all matching items in Address Manager across all configurations.

Use this method to fetch response policy items and their associated properties or objects such as *Response Policy*, *RP Zone*, *Threat Protection category for feed data*, *policy type* and *the parent configuration*.

Output / Response

Returns an array of `ResponsePolicySearchResult` objects. Each object contains information of one Response Policy item found either in local Response Policies or BlueCat Security feed data.

API Call:

```
ResponsePolicySearchResult[] searchResponsePolicyItems( String keyword, String scope, int start, int count, String properties )
```

Parameter	Description
keyword	<p>The search string for which you wish to search.</p> <ul style="list-style-type: none"> • ^—matches the beginning of a string. For example: ^ex matches example but not text. • \$—matches the end of string. For example: ple\$ matches example but not please. • *—matches zero or more characters within a string. For example: ex*t matches exit and excellent.
scope	<p>The scope in which the search is to be performed. The possible values are:</p> <ul style="list-style-type: none"> • RPItemSearchScope.LOCAL—to search policy items configured in local Response Policies. • RPItemSearchScope.FEED—to search policy items configured in predefined BlueCat Security Feed data.

Parameter	Description
	<ul style="list-style-type: none"> RPItemSearchScope.ALL—to search policy items configured in both local Response Policies and predefined BlueCat Security Feed data.
start	A starting number from where the search result will be returned. The possible value is a positive integer ranging from 0 to 999. For example, specifying 99 will return the search result from the 100th result to the maximum number that you specify with the <i>count</i> option.
count	The total number of results to be returned. The possible value is a positive integer ranging from 1 to 1000.
properties	Reserved for future use. Use an empty string ("") for now.

Reverse zone name format

Address Manager creates the reverse zone and reverse zone structure automatically when deploying a DNS deployment role configuration to DNS/DHCP Server.

Previously, only one reverse zone format was supported and this resulted in issues when importing an existing reverse zone that did not follow the default Address Manager format. Because there are number of other acceptable formats that can be used to generate reverse DNS zones, Address Manager now supports setting a custom reverse zone format.

Add Reverse Zone Name Format

Adds custom DNS reverse zone name formats.

Output / Response

Returns the object ID of the new DNS reverse zone name format added.

API Call:

```
long addDNSDeploymentOption( long entityId, String name, String value, String properties )
```

Parameter	Description
entityId	The object ID for the entity to which this deployment option is being added.
name	The name must be <i>DNSOptions.CLASSLESS_REVERSE_ZONE_FORMAT</i> or <i>"classless-reverse-zoneformat"</i> .
value	<p>The following values are supported:</p> <ul style="list-style-type: none"> <i>ReverseZoneFormatType.STARTIP_NETMASK_NET</i> or "[start-ip]-[netmask].[net].inaddr.arpa" <i>ReverseZoneFormatType.STARTIP_ENDIP_NET</i> or "[start-ip]-[end-ip].[net].in-addr.arpa" <i>ReverseZoneFormatType.STARTIP_SLASH_NETMASK_NET</i> or "[start-ip]/[netmask].[net].in-addr.arpa" <i>ReverseZoneFormatType.STARTIP_SLASH_ENDIP_NET</i> or "[start-ip]/[end-ip].[net].inaddr.arpa" <i>ReverseZoneFormatType.CUSTOM</i> + <i>ReverseZoneFormatType.SEPARATOR</i> + "[customformat-value]" or <i>"custom:<custom-format-value>"</i>
properties	Adds object properties, including comments and user-defined fields.

Deployment options

Deployment is the process by which the configuration in Address Manager becomes a running set of services on Address Manager-managed servers, and Deployment Options define the deployment of Address Manager DNS and DHCP services.

Deployment options can be applied at many different levels within a configuration such as server, block, network, DNS Views, or DNS Zones level.

Getting deployment options

This is the generic API method for getting Deployment options for Address Manager DNS and DHCP services.

Get Deployment Options

Retrieves deployment options for Address Manager DNS and DHCP services.

Output / Response

Returns all deployment options assigned to the specified object including inherited options from higher-level parent objects. If an option is inherited and overridden, then only the overriding option will be returned.



Note: Multiple raw options with long values can cause longer than normal processing time when returning the values.

API Call:

```
APIDeploymentOption[] getDeploymentOptions( long entityId, String optionTypes, long serverId )
```

Parameter	Description
entityId	The object ID of the entity to which the DNS or DHCP deployment option is assigned.
optionTypes	<p>The type of deployment options. Multiple options can be separated by a (pipe) character. This value must be one of the following items:</p> <ul style="list-style-type: none"> • DNSOption • DNSRawOption • DHCPRawOption • DHCPV6RawOption • DHCPV4ClientOption • DHCPV6ClientOption • DHCPServiceOption • DHCPV6ServiceOption • VendorClientOption • StartOfAuthority <p>For complete list of Option Types and Object Types constants, refer to Option Types on page 211 and Object Types on page 209.</p> <ul style="list-style-type: none"> • If Invalid deployment option types or invalid strings are specified, the API execution will fail and return the error message: <i>"Invalid deployment option found"</i>. For example, if the user passes DHCPv6ClientOption for IPv4 networks, it will return this error message as DHCPv6 client options are not a valid for IPv4 networks.

Parameter	Description
	<ul style="list-style-type: none"> If specified as an <i>empty string</i> (""), all deployment options for the specified entity will be returned. Depending on the type of DNS deployment option being retrieved, the format of the value might differ. For more information, refer to Reference: DNS Option value formats on page 138.
serverId	<p>The specific server or server group to which options are deployed. The valid values are as follows:</p> <ul style="list-style-type: none"> >0—returns only the options that are linked to the specified server ID. <0—returns all options regardless of the server ID specified. =0—returns only the options that are linked to all servers.

Raw deployment option

This is the generic API method for adding and updating raw deployment option to DNS and DHCP services in a raw format.

The raw option allows you to add DNS or DHCP options that are not directly supported by Address Manager.

A raw option is passed to the DNS or DHCP service on the managed server exactly as you type it in the **rawData** parameter. Therefore, it is essential that you enter the data with the correct syntax. There is no error checking or data checking on the raw option .

Add Raw Deployment Option

Adds deployment options to DNS or DHCP services in a raw format that will be passed to the service when deployed.

Output / Response

Returns the object ID for the newly added Raw option.

API Call:

```
long addRawDeploymentOptions( long parentId, String optionTypes, String rawData, String properties )
```

Parameter	Description
parentId	The object ID of the entity to which the DNS or DHCP raw deployment option is assigned.
optionType	<p>The type of the raw deployment option being assigned. This value must be one of the following items:</p> <ul style="list-style-type: none"> DNS_RAW DHCP_RAW DHCPV6_RAW <p>For complete list of Option Types constants, refer to Option Types on page 211.</p>
rawData	The raw option value. The maximum supported characters are 65,536. The raw option will be passed to the DNS or DHCP service on the managed server exactly as you enter here. Therefore, it is essential that you enter the data with the correct syntax.

Parameter	Description
properties	Adds object properties, including associated server and server group, and user-defined fields.

Update Raw Deployment Option

Updates raw deployment options.

Output / Response

None.

API Call:

```
long updateRawDeploymentOptions( APIDeploymentOption option )
```

Parameter	Description
option	The DNS or DHCP raw option object to be updated.

TFTP

Address Manager uses Trivial File Transfer Protocol (TFTP) to provide configuration files to end-point devices.

TFTP Groups

TFTP files are organized into a list of tree structures. Each tree has a root, called a TFTP group. The leaves of this tree are files, and the nodes of the tree are folders. TFTP groups are child objects of configurations. Each tree structure reflects the directory structure on a target TFTP server.

Add TFTP Group

Adds TFTP groups.

Output / Response

Returns the object ID for the new TFTP group.

API Call:

```
long addTFTPGroup( long configurationId, String name, String properties )
```

Parameter	Description
configurationId	The object ID of the configuration to which the TFTP group is being added.
name	The name of the TFTP group.
properties	Adds object properties, including comments and user-defined fields.

Update TFTP Group

A TFTP group's **name** property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

TFTP Group Generic Methods

TFTP groups use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

TFTP Folders

TFTP folders are used to create the directory structure on the TFTP server.

Add TFTP Folder

Adds TFTP folders.

Output / Response

Returns the object ID for the new TFTP folder.

API Call:

```
long addTFTPFolder( long parentId, String name, String properties )
```

Parameter	Description
parentId	The object ID of the parent object of the TFTP folder. The parent is either a TFTP group or another TFTP folder object.
name	The name of the TFTP folder.
properties	Adds object properties, including comments and user-defined fields.

Update TFTP Folder

A TFTP folder's **name** property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

TFTP Folder Generic Methods

TFTP folders use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

TFTP Files

TFTP files contain configuration information that is passed to the client end-point devices to be configured.

Add TFTP File

Adds TFTP files.

Output / Response

Returns the object ID for the new TFTP file.

API Call:

```
long addTFTPFile( long parentId, String name, String version, byte[] data, String properties )
```

Parameter	Description
parentId	The object ID of the parent object of the TFTP file. The parent will always be a TFTP folder.
name	The name of the TFTP file.
version	The version of the file. This parameter is optional.

Parameter	Description
data	The file to be uploaded and distributed to clients by TFTP. The file is passed to Address Manager as a byte array.
properties	Adds object properties, including comments and user-defined fields.

Update TFTP File

A TFTP file's **name**, **version**, and **description** properties can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

TFTP File Generic Methods

TFTP files use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Servers and Deployment

Address Manager deploys settings and services separately from designing and configuring the actual services.

Server objects must be added to Address Manager, and then server roles can associate various DNS, DHCP, and TFTP services to the servers on which they will run. Deployment can be scheduled within the Address Manager interface, but only immediate deployments can be performed through the Address Manager API.

Servers

To use a DNS/DHCP Server with Address Manager, you must add the server to the Address Manager configuration.

This involves providing information in the Add Server screen, and then connecting to the server, or using the `addServer()` method. A successful connection places the DNS/DHCP Server under the control of Address Manager and disables the native DNS/DHCP Server command server agent. As a result, the server is no longer managed through the DNS/DHCP Server Management Console and responds only to commands from Address Manager.

Servers are added to a Address Manager configuration so that services can be deployed to them using deployment roles and server options. Servers can be added with the API or in the Address Manager GUI without connecting to them if the server objects need to be created and configured before the actual servers are available. The method described here adds the server to a Address Manager configuration only, and does not connect to the server.

Before deploying a configuration, you must connect to servers added using this method or using the Address Manager web interface. For more information about connecting to existing servers using the Address Manager web interface refer to the Address Manager Administration Guide and the online help. Server control is available through the Address Manager web interface and the Address Manager Administration Console.

Address Manager v8.0.0 and greater, and DNS/DHCP Server v7.0.0 and greater feature support for dedicated management on multi-interface DNS/DHCP Server appliances (DNS/DHCP Server hardware models 1900, 1925, and 1950). These DNS/DHCP Server appliances include support for three interface ports (Services, XHA, Management) and four interface ports (Services, XHA, Management, and Redundancy through port bonding: eth0 + eth3).

- ➔ **Note:** The procedure for configuring a DNS/DHCP Server and adding it to Address Manager will vary according to the number of interfaces on your DNS/DHCP Server appliance, and the number of interfaces that you wish to utilize.

The following table describes the interfaces that are being used by different types of DNS/DHCP Server:

Number of ports	eth0	eth1	eth2	eth3
2	Services / Management	XHA	N/A	N/A
3	Services	XHA	Dedicated Management	N/A
4	Services	XHA	Dedicated Management	Redundancy

- ➔ **Note:** If you are using multi-port DNS/DHCP Server appliances and want to use dedicated management, you must enable it from the Administration Console before adding a DNS/DHCP Server to Address Manager.

Add Server

Adds servers to Address Manager.

- ➔ **Note:** Existing customers who have upgraded their Address Manager API to v8.0.0 or greater may need to update their API calls to add a server with dedicated management enabled. For more information, refer to [KB-939](#).


Output / Response

Returns the object ID for the new server.

API Call:

```
long addServer ( long configurationId, string name, string defaultInterfaceAddress, string fullHostName, string profile, string properties )
```

Parameter	Description
configurationId	The object ID of the configuration to which the server is being added.
name	The name of the server within Address Manager.
defaultInterfaceAddress	The physical IP address for the server within Address Manager.
fullHostName	The DNS FQDN by which the server is referenced.
profile	The server capability profile. The profile describes the type of server or appliance being added and determines the services that can be deployed to this server. This must be one of the constants found in Server Capability Profiles on page 213.
properties	A string containing the following options: <ul style="list-style-type: none"> • connected—either <i>true</i> or <i>false</i>; indicates whether or not to connect to a server. In order to add and configure multi-port DNS/DHCP Servers, this option must be set to true. If false, other interface property options will be ignored. • upgrade—indicates whether or not to apply the latest version of DNS/DHCP Server software once the appliance is under Address Manager control. The value is either <i>true</i> or <i>false</i> (by default, <i>true</i>). • password—the server password (by default, bluecat).

Parameter	Description
	<ul style="list-style-type: none"> • servicesIPv4Address—IPv4 address used only for services traffic such as DNS, DHCP, DHCPv6, and TFTP. If dedicated management is enabled, this option must be specified. • servicesIPv4Netmask—IPv4 netmask used only for services traffic such as DNS, DHCP, DHCPv6, and TFTP. If dedicated management is enabled, this option must be specified. • servicesIPv6Address—IPv6 address used only for services traffic such as DNS, DHCP, DHCPv6, and TFTP. This is <i>optional</i>. • servicesIPv6Subnet—IPv6 subnet used only for services traffic such as DNS, DHCP, DHCPv6, and TFTP. This is <i>optional</i>. • xhaIPv4Address—IPv4 address used for XHA. This is <i>optional</i>. • xhaIPv4Netmask—IPv4 netmask used for XHA. This is <i>optional</i>. • redundancyScenario—networking redundancy scenarios. The possible values are <i>ACTIVE_BACKUP (Failover)</i> and <i>IEEE_802_3AD (Load Balancing)</i>. <p>The following properties only apply to the Windows server:</p> <ul style="list-style-type: none"> • ProteusDDW—enter the object ID of the DDW server. • readOnly—indicates if the Windows server is added in read-only mode. The value is either <i>true</i> or <i>false</i> (by default, <i>true</i>). If set to <i>false</i>, the server will be added in read-write mode. • enableDNS—indicates if DNS is enabled. The value is either <i>true</i> or <i>false</i> (by default, <i>true</i>). • enableDHCP—indicates if DHCP is enabled. The value is either <i>true</i> or <i>false</i> (by default, <i>true</i>). • importViewName—enter a View name for the Windows server. When you manage Windows DNS from Address Manager, you must specify a DNS View. Only DNS records contained in this View are deployed. This is mandatory if enableDNS is set to <i>true</i>. • authenticationCredentialDomain—enter the domain name of the Windows Active Directory domain to which this server belongs. If the server is not a member of a domain, enter the server's NETBIOS computer name. • authenticationCredentialUsername—enter the username for the Windows server. • authenticationCredentialPassword—enter the user password for the Windows server. <p> Note: For DNS/DHCP Servers without multi-port support, the interface-related property options will be ignored.</p>

Import Server


Imports Windows DNS or DHCP services from Managed Windows servers.

Output / Response

Returns void.

API Call:

```
void importServer( long serverId, boolean importDns, boolean importDhcp, string properties
)
```

Parameter	Description
serverId	The object ID of the server that needs to be imported.  Note: You must set at least one of the following Boolean parameters, but you cannot set both of them to False .
importDns	Imports DNS service if true.
importDhcp	Imports DHCP service if true.
properties	Reserved for future use.

Replace Server

Allows you to replace a server.



Output / Response

Replaces the server using the existing server ID.

API Call:

```
void replaceServer( long serverId, string name, string defaultInterface, string hostName,
string password, boolean upgrade, string properties )
```

Parameter	Description
serverId	The object ID of the server that needs to be replaced.
name	Name of the server to be replaced.
defaultInterface	Management interface address for the server.
hostName	The DNS FQDN by which the server is referenced.
password	The server password (by default, bluecat).
upgrade	Flag indicating that server needs to be upgraded or not. True means server needs to be upgraded.
properties	A string containing the following options: <ul style="list-style-type: none"> • servicesIPv4Address—IPv4 address used only for services traffic such as DNS, DHCP, DHCPv6 and TFTP. If <i>dedicated management</i> is enabled, this option must be specified. If <i>dedicated management</i> is disabled, this address must be the same as defaultInterfaceAddress which is management interface address. • servicesIPv4Netmask—IPv4 netmask used only for services traffic such as DNS, DHCP, DHCPv6 and TFTP. If <i>dedicated management</i> is enabled, this option must be specified. If <i>dedicated management</i> is disabled, this netmask address must be the same as the management interface netmask address. • servicesIPv6Address—IPv6 address used only for services traffic such as DNS, DHCP, DHCPv6 and TFTP. This is <i>optional</i>. • servicesIPv6Subnet—IPv6 subnet used only for services traffic such as DNS, DHCP, DHCPv6 and TFTP. This is <i>optional</i>. • xhaIPv4Address—IPv4 address used for XHA. This is <i>optional</i>. • xhaIPv4Netmask—IPv4 netmask used for XHA. This is <i>optional</i>. • redundancyScenario—networking redundancy scenarios. The possible values are ACTIVE_BACKUP (<i>Failover</i>) and IEEE_802_3AD (<i>Load Balancing</i>).

Parameter	Description
	<ul style="list-style-type: none"> • resetServices—allows you to replace the DNS/DHCP Server while maintaining existing configurations for DNS, DHCP, and TFTP services. Define this option only if you have modified the IPv4 or IPv6 addresses of the Services interface or wish to reset configurations for DNS, DHCP, and TFTP services on the DNS/DHCP Server. The value is either true or false (by default, false). <p> Note: For DNS/DHCP Servers without multi-port support, the interface-related property options will be ignored.</p> <p> Caution: Resetting DNS/DHCP Servers will result in a service outage. This service outage will last until you have deployed services to the replacement system. Only reset DNS/DHCP Server services if you are replacing the DNS/DHCP Server with a new appliance of a different type and/or reconfiguring the IPv4 or IPv6 addresses of the Services interface on the appliance. BlueCat recommends that you schedule a maintenance window before performing a reset of DNS/DHCP Server services.</p>

Deploy Server

Deploys servers. When this method is invoked, the server is immediately deployed.

Output / Response

None.

Deployment is the process through which the configuration created in Address Manager becomes a running set of services on the Address Manager-managed servers. Deployment takes account of the IP, DHCP, and DNS design determined during configuration. This is represented by a set of service configuration files deployed to the servers.

API Call:

```
void deployServer( long serverId )
```

Parameter	Description
serverId	The object ID of the server to be deployed.

Deploy Server Configuration

Allows you to deploy specific configuration(s) to a particular server.

Output / Response

Deploys specific configuration(s) to a particular server.

API Call:

```
void deployServerConfig( long serverId, String properties )
```

Parameter	Description
serverId	The database object ID of the server that will immediately be deployed.
properties	A string containing property names. The property names available in the ObjectProperties are <i>ObjectProperties.services</i> , and

Parameter	Description
	<p><i>ObjectProperties.forceDNSFullDeployment</i>. Multiple options can be separated by a (pipe) character. For example:</p> <pre>ObjectProperties.services=DNS forceDNSFullDeployment=true</pre> <p>The values for properties are:</p> <ul style="list-style-type: none"> • services—the name of the valid service configuration that needs to be deployed. These are the valid values for the services: DNS, DHCP, DHCPv6, and TFTP. • forceDNSFullDeployment—a boolean value. set to true to perform a full DNS deployment. Omit this parameter from the method call to perform a differential deployment.

Deploy Server Services

Allows you to deploys specific service(s) to a particular server.

Output / Response

Deploys specific service(s) to a particular server.

API Call:

```
void deployServerServices( long serverId, String services )
```

Parameter	Description
serverId	The database object ID of the server for which deployment services to be deployed.
services	The name of the valid services to be deployed. Specify multiple services using a comma (.). The valid format is <code>services=DNS,DHCP,TFTP</code> .

Quick Deployment

Allows you to instantly deploy changes you made to DNS resource records made since the last full or quick deployment. This function applies only to DNS resource records that you have changed and does not deploy any other data.

Output / Response

Instantly deploys changes to DNS resource records made since the last full deployment or quick deployment.

API Call:

```
void quickDeploy( long entityId, String properties )
```

Parameter	Description
entityId	The object ID of the DNS zone or network for which deployment service needs to be deployed.
properties	<p>A string containing the services option. It can also be <i>empty</i>.</p> <ul style="list-style-type: none"> • services—the name of the valid service that need to be deployed. The <i>only</i> valid service name for quick deployment is DNS. Any other service names will throw an error.

Deployment Status

Returns the server's deployment status.

Output / Response

Returns status code for deployment of a particular server. These are the possible returning code values:

- EXECUTING = -1
- INITIALIZING = 0
- QUEUED = 1
- CANCELLED = 2
- FAILED = 3
- NOT_DEPLOYED = 4
- WARNING = 5
- INVALID = 6
- DONE = 7
- NO_RECENT_DEPLOYMENT = 8

API Call:

```
int getServerDeploymentStatus( long serverId, String properties )
```

Parameter	Description
serverId	The object ID of the server whose deployment status needs to be checked.
properties	Ignore this for now. The valid value is <i>empty</i> .

Server Generic Methods

Servers use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Get Published Interface

To get the published interface IP address for a server, use the `getEntities()` method with the **PublishedServerInterface** type to return the server properties. The **publiushedInterfaceAddress** property appears in the returned properties string.

Server Group

A Server Group is a logical container in which multiple servers are grouped together for common purposes.

Grouping servers in a Server Group allows you to apply DNS and DHCP deployment options to all servers that comprise the group. After you have created a Server Group, you can add one or more DNS/DHCP Servers to the Server Group. DNS or DHCP options can then be applied to the Server Group and these options will be inherited by all servers that are added to that specific Server Group. When DNS or DHCP options are removed from the Server Group, the options are no longer inherited by the servers that are added to that group.



Attention:

- Only BlueCat DNS/DHCP Servers can be added to a Server Group.
- A DNS/DHCP Server can only be added to a single server group.
- Deployment options applied to specific servers within a Server Group will override the options set on the Server Group.

Add Server Group

Adds a Server Group that will contain multiple DNS/DHCP Servers.

You can use the generic `addEntity()` method to add a Server Group. Use **ObjectType.ServerGroup** to define the Server Group entity. For example:

```
serverGroup = new APIEntity( 0, name, properties, ObjectTypes.ServerGroup );
serverGroupId = service.addEntity( configurationId, serverGroup );
```

➔ **Note:** When defining a Server Group entity, the entity name cannot be empty.

For more information about `addEntity()`, refer to [Adding Objects](#) on page 31

Output / Response

Returns the object ID for the new Server Group.

Update Server Group

Updates the Server Group's **name** property using the generic `update()` method.

When updating a Server Group entity, the **name** property must not be empty. For more information, refer to [Updating Objects](#) on page 50.

Server Group Generic Methods

Server Group uses the generic `get()` and `delete()` methods for entities.

For more information, refer to [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Add Server to Server Group

Use the generic `linkEntities()` method to add a server to a Server Group.

For more information, refer to [Link Entities](#) on page 53.

Remove Server from Server Group

Use the generic `unlinkEntities()` method to remove a server from a Server Group.

For more information, refer to [Unlink Entities](#) on page 54.

DNS and DHCP Deployment Roles

Deployment roles determine the general pattern of the deployment. A deployment role exists on a particular server interface (physical or published) specified with an IP address.

Each server interface can have multiple DNS roles and one DHCP role with the most locally-specified server role taking precedence. The addition of a deployment role is allowed only if that role is possible under that server's service capability profile as described in the *Address Manager Administration Guide*.

Get Servers Associated with a Deployment Role

Returns a list of all servers associated with the specified deployment role.

Output / Response

Returns an `APIEntity` object representing the servers associated with the specified deployment role.

API Call:

```
APIEntity getServerForRole( long roleId )
```

Parameter	Description
roleId	The object ID for the deployment role whose servers are to be returned.

Get Server's Associated Deployment Roles

Returns a list of all deployment roles associated with the server.

Output / Response

Returns a list of all deployment roles associated with the server.

API Call:

```
APIDeploymentRole[] getServerDeploymentRoles( long serverId )
```

Parameter	Description
serverId	The object ID of the server with which deployment roles are associated.

Get Deployment Roles for DNS and IP Address Space Objects

Returns the DNS and DHCP deployment roles associated with the specified object. For DNS Views and zones, `getDeploymentRoles()` returns DNS deployment roles. For IP address space objects, such as IPv4 blocks and networks, IPv6 blocks and networks, DHCP classes, and MAC pools, `getDeploymentRoles()` returns DNS and DHCP deployment roles.

Output / Response

Returns an array of `APIDeploymentRole` objects representing the deployment roles associated with the specified object. The properties string contains the following elements:

- **view**—for DNS deployment roles set for IP address space objects.
- **zoneTransServerInterface**—the server interface for zone transfers for the deployment role types of slave, stealth slave, forwarder and stub.
- **inherited**—returns *true* or *false* to indicate whether the deployment role was inherited or not.

API Call:

```
APIDeploymentRole[] getDeploymentRoles( long entityId )
```

Parameter	Description
entityId	The object ID for a DNS view, DNS zone, IPv4 block or network, IPv6 block or network, DHCP class, or MAC pool.

Move Deployment Roles

Moves all DNS and DHCP deployment roles from a server to the specified interface of another server.

- ➔ **Note:** You **CANNOT** move deployment roles if the target server has deployment roles associated with it. You **MUST** remove all deployment roles assigned to the target server before moving the roles.
- ➔ **Note:** Either the **moveDnsRoles** or **moveDhcpRoles** parameter must be set to true.

API Call:

Output / Response

None.

```
void moveDeploymentRoles( long sourceServerId, long targetServerInterfaceId, boolean moveDnsRoles, boolean moveDhcpRoles, String options )
```

Parameter	Description
sourceServerId	The object ID of the server that contains the roles.
targetServerInterfaceId	The object ID of the server interface of the server to which the roles are to be moved.
moveDnsRoles	If set to <i>true</i> , DNS roles will be moved to the target server interface.
moveDhcpRoles	If set to <i>true</i> , DHCP roles will be moved to the target server interface.
options	This is reserved for future use.

DHCP Deployment Roles

The DHCP server role can be set to either **master** or **none**. Roles set to **none** are not deployed.

Roles can also be applied at many points throughout a configuration, with the most local roles taking precedence over those assigned to objects higher in the object hierarchy.

Add DHCP Deployment Role

Adds a DHCP deployment role to a specified object.

Output / Response

Returns the object ID for the new DHCP server role object.

API call:

```
long addDHCPDeploymentRole( long entityId, long serverInterfaceId, String type, String properties )
```

Parameter	Description
entityId	The object ID for the object to which the deployment role is to be added.
serverInterfaceId	The object ID of the server interface to which the role is to be deployed.
type	The type of DHCP role to be added. The type must be one of those listed in DHCP Deployment Role Types on page 195.
properties	A string containing options including: <ul style="list-style-type: none"> inherited—either true or false; indicates whether or not the deployment role was inherited. secondaryServerInterfaceId—the object ID of the secondary server interface for a DHCP failover.

Get DHCP Deployment Role

Retrieves the DHCP deployment role assigned to a specified object.

Output / Response

Returns the DHCP deployment role assigned to the specified object, or returns an empty **APIDeploymentRole** if no role is defined. For information about the output properties, refer to [Property Options Reference](#) on page 241.

API call:

```
APIDeploymentRole getDHCPDeploymentRole( long entityId, long serverInterfaceId )
```

Parameter	Description
entityId	The object ID for the object to which the deployment role is assigned.
serverInterfaceId	The object ID of the server interface to which the role is assigned.

Update DHCP Deployment Role

Updates a DHCP deployment role.

Output / Response

None.

```
void updateDHCPDeploymentRole( APIDeploymentRole role )
```

Parameter	Description
role	The DHCP deployment role object to be updated.

Delete DHCP Deployment Role

Deletes DHCP deployment roles.

Output / Response

None.

```
void deleteDHCPDeploymentRole( long entityId, long serverInterfaceId )
```

Parameter	Description
entityId	The object ID for the object from which the deployment role is to be deleted.
serverInterfaceId	The object ID of the server interface from which the deployment roles is to be deleted.

DNS Deployment Roles

At a minimum, DNS roles must be applied at the View level in order for DNS deployment to occur.

They can also be applied further into the DNS core if desired. For Reverse DNS, a DNS deployment role must be applied to either a block or a network in order to deploy the Reverse DNS settings for that object and its sub-objects.

The following DNS server roles are available:

DNS role	Description
None	This DNS role is not deployed. Use this option for DNS objects that exist, but should not be deployed.
Master	This role deploys details and options consistent with a DNS master. This role is also used on a DNS/DHCP Server 250 with the appropriate DNS options to create a caching-only DNS server.
Hidden Master	This role deploys details and options consistent with a DNS master. However, no name server records are created for the server, thus hiding it from DNS queries.
Slave	This role deploys details and options consistent with a DNS slave.

DNS role	Description
Stealth Slave	A stealth slave is a DNS slave server that does not have any name server records pointing to it. This is useful for testing purposes or for having a hot spare stand-by server. However, this is not a commonly used DNS role.
Forwarder	This role deploys details and options consistent with a DNS forwarder. You must use both the forwarding policy and forwarding options to make this role function properly.
Stub	A stub zone contains only the name server records for a domain. Address Manager generates name server records automatically during deployment, so a zone deployed within a stub role will not contain any user-selected details or options.
Recursion	This role creates DNS caching servers. The options and root zone associated with this role are described in the <i>Address Manager Administration Guide</i> .

Add DNS Deployment Role

Adds a DNS deployment role to a specified object.

Output / Response

Returns the object ID for the new DNS server role object.

API call:

```
long addDNSDeploymentRole( long entityId, long serverInterfaceId, String type, String properties )
```

Parameter	Description
entityId	The object ID for the object to which the deployment role is to be added.
serverInterfaceId	The object ID of the server interface to which the role is to be added.
type	The type of DNS role to be added. The type must be one of those listed in DNS Deployment Role Type on page 197.
properties	Adds object properties, including the View associated with this DNS deployment role and user-defined fields.

Get DNS Deployment Role

Retrieves a DNS deployment role from a specified object.

Output / Response

Returns a DNS deployment role from the specified object, or returns an empty **APIDeploymentRole** if no role is defined. For information about the output properties, refer to [Property Options Reference](#) on page 241.

API call:

```
APIDeploymentRole getDNSDeploymentRole( long entityId, long serverInterfaceId )
```

Parameter	Description
entityId	The object ID for the object to which the DNS deployment role is assigned.
serverInterfaceId	The object ID of the server interface to which the DNS deployment role is assigned.

Get DNS Deployment Role for View

Retrieves the DNS deployment role assigned to a view-level objects in the IP space for ARPA zones.

Output / Response

Returns the requested APIDeploymentRole object. For information about the output properties, refer to [Property Options Reference](#) on page 241.

API call:

```
APIDeploymentRole getDNSDeploymentRoleForView( long entityId, long
serverInterfaceId, long viewId )
```

Parameter	Description
entityId	The object ID for the object to which the DNS deployment role is assigned.
serverInterfaceId	The object ID of the server interface to which the DNS deployment role is assigned.
viewId	The view in which the DNS deployment role is assigned.

Update DNS Deployment Role

Updates a specified DNS deployment role.

Output / Response

None.

API call:

```
void updateDNSDeploymentRole( APIDeploymentRole role )
```

Parameter	Description
role	The DNS deployment role object to be updated.

Delete DNS Deployment Role

Deletes a specified DNS deployment roles.

Output / Response

None.

API call:

```
void deleteDNSDeploymentRole( long entityId, long serverInterfaceId )
```

Parameter	Description
entityId	The object ID for the object from which this DNS deployment role is to be deleted.
serverInterfaceId	The object ID of the server interface to which the DNS deployment role is assigned.

Delete DNS Deployment Role for View

Deletes the DNS deployment role assigned to view-level objects in the IP space for ARPA zones.

Output / Response

None.

API call:

```
void deleteDNSDeploymentRoleForView( long entityId, long serverInterfaceId, long viewId )
```

Parameter	Description
entityId	The object ID for the object from which this DNS deployment role is to be deleted.
serverInterfaceId	The object ID of the server interface to which the DNS deployment role is assigned.
viewId	The view from which the DNS deployment role is to be deleted.

TFTP Deployment Roles

TFTP deployment roles are used to assign TFTP services to DHCP servers.

Add TFTP Deployment Role

Adds a TFTP deployment role to a specified object.

Output / Response

Returns the object ID for the new TFTP deployment role object.

API call:

```
long addTFTPDeploymentRole( long entityId, long serverId, String properties )
```

Parameter	Description
entityId	The object ID for the object to which the TFTP deployment role is to be added.
serverId	The object ID of the server interface to which the TFTP deployment role is to be added.
properties	Adds object properties, including user-defined fields.

Update TFTP Deployment Role

TFTP deployment roles cannot be updated.

TFTP Deployment Role Generic Methods

TFTP deployment roles use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Crossover High Availability (xHA)

DNS/DHCP Server Crossover High Availability (xHA) provides disaster recovery through the use of redundant appliances: xHA makes two DNS/DHCP Server appliances function as a single appliance.

If one of the appliances fails for any reason, the other takes its place and continues providing services. The pair appears as a single server for DNS queries because both servers share an IP address. Each server in the pair has its own IP addresses for management through Address Manager. For details about xHA, please refer to *Address Manager Administration Guide*.

Requirements for creating an xHA pair

Before you create an xHA pair in Address Manager, you must have the following requirements in place:

- You must have at least two connected and managed DNS/DHCP Servers in the configuration.
- DNS/DHCP Servers must be either two physical appliances or two virtual machines. Mixed xHA pairs of appliance and VM are **NOT** supported.
- Both DNS/DHCP Servers must be at the **same software version** before creating an xHA pair.
- Both DNS/DHCP Servers must be of the **same profile**, such as two DNS/DHCP Server 60 or two DNS/DHCP Server 100 profiles.
- Both DNS/DHCP Servers must be of the **same architecture**, That is, two 64-bit servers, or two 32-bit servers (such as two XMB2 appliances).



Attention: Cross-architecture xHA pairs, such as one 64-bit node and one 32-bit node, are **NOT** supported.

- In order to create an xHA pair with the Active node on which the dedicated management interface enabled, the dedicated management interface on the Passive node must be enabled.
- The Active and Passive nodes must be on the same network.
- The servers for the xHA pair must not be associated with a deployment schedule.
- The server intended for the passive role must not be associated with a deployment role.
- To avoid split-brain scenarios (where both servers are active or passive at the same time), the use of xHA Backbone Communication is mandatory.



Attention:

- If you are currently using the xHA/eth1 ports for another purpose, you can reset and then reconfigure them for xHA communication, but you cannot use the eth1 ports for xHA communication and for their previous purpose.
- If you are upgrading from a previous version of DNS/DHCP Server, you must delete each eth1 port to reset it. This is because previous versions did not support eth1, and it is not reset automatically.

Creating an xHA

With xHA prerequisites are met, you can create an xHA pair.



Note: You cannot configure interface and network settings of DNS/DHCP Server appliances that are part of a functioning xHA pair. You must configure interface and network settings before creating a xHA pair.

Create xHA

Creates an xHA pair.


Output / Response

Returns the object ID for the xHA pair created.

API call:

```
long createXHAPair( long configurationId, long activeServerId, long passiveServerId,
String activeServerNewIPv4Address, String properties )
```

Parameter	Description
configurationId	The object ID of the configuration in which the xHA servers are located.
activeServerId	The object ID of the active DNS/DHCP Server server.
passiveServerId	The object ID of the passive DNS/DHCP Server server.

Parameter	Description
activeServerNewIPv4Address	The new IPv4 address for the active server.  Note: This is the physical interface of the active server used during creation of the pair. The original IP address of the active server is assigned to the virtual interface.
properties	A string containing options listed in List of options .

List of options

activeServerPassword	The deployment password for the active server (by default, <i>bluecat</i>).
passiveServerPassword	The deployment password for the passive server (by default, <i>bluecat</i>).
pingAddress	An IPv4 address that is accessible to both active and passive servers in the xHA pair.
ip6Address	An optional IPv6 address for the xHA pair.
newManagementAddress	The new IPv4 address for the Management interface for the active server (only for DNS/DHCP Servers with dedicated management enabled).
backboneActiveServerIPv4Address	The IPv4 address of the xHA interface for the active server (eth1).
backboneActiveServerIPv4Netmask	The IPv4 netmask of the xHA interface for the active server (eth1).
backbonePassiveServerIPv4Address	The IPv4 address of the xHA interface for the passive server (eth1).
backbonePassiveServerIPv4Netmask	The IPv4 netmask of the xHA interface for the passive server (eth1).
activeServerIPv4AddressForNAT	The inside virtual IPv4 address for the active server.
passiveServerIPv4AddressForNAT	The inside virtual IPv4 address for the passive server.
activeServerNewIPv4AddressForNAT	The inside physical IPv4 address for the active server.

Edit xHA

Updates the xHA pair created.

Output / Response

None.

API call:

```
void editXHAPair( long xHAServerId, String name, String properties )
```

Parameter	Description
xHAServerId	The object ID of the xHA server.
name	The name of the xHA server being updated.
properties	A string containing options listed in List of options .

List of options

backboneActiveServerIPv4Address	The IPv4 address of the xHA interface for the active server (eth1).
backboneActiveServerIPv4Netmask	The IPv4 netmask of the xHA interface for the active server (eth1).
backbonePassiveServerIPv4Address	The IPv4 address of the xHA interface for the passive server (eth1).
backbonePassiveServerIPv4Netmask	The IPv4 netmask of the xHA interface for the passive server (eth1).
overrideDHCPValidation	<i>True</i> or <i>false</i> ; indicates whether or not the deployment validation settings set at the configuration level is inherited.
checkDHCPConfigurationDeployment	<i>True</i> or <i>false</i> ; checks the syntax of the <i>dhcpd.conf</i> file and validate data deployed from Address Manager.
overrideDNSValidation	<i>True</i> or <i>false</i> ; indicates whether or not the deployment validation settings set at the configuration level is inherited.
checkDNSConfigurationDeployment	<i>True</i> or <i>false</i> ; checks the syntax of the <i>named.conf</i> file and validate data deployed from Address Manager.
checkDNSZonesDeployment	<i>True</i> or <i>false</i> ; checks the syntax of each DNS zone file and validate data deployed from Address Manager.
postLoadZoneIntegrityValidationDNSDeploy	<p>Checks the syntax based on the mode selected. The available modes are as follows:</p> <ul style="list-style-type: none"> • Full—checks for the following conditions: <ul style="list-style-type: none"> • If MX records refer to A or AAAA records, for both in-zone and out-of-zone hostnames. • If SRV records refer to A or AAAA records, for both in-zone and out-of-zone hostnames. • If Delegation NS records refer to A or AAAA records, for both in-zone and out-of-zone hostnames • If glue address records in the zone match those specified by the child. • Local—checks for the following conditions: <ul style="list-style-type: none"> • If MX records refer to A or AAAA records, for in-zone hostnames. • If SRV records refer to A or AAAA records, for in-zone hostnames. • If Delegation NS records refer to an A or AAAA record, for in-zone hostnames. • If glue address records in the zone match those specified by the child. • Full-sibling—performs the same checks as in Full mode but does not check the glue records.

	<ul style="list-style-type: none"> • Local-sibling—performs the same checks as in Local mode but does not check the glue records. • None—disables all post-load zone integrity checks.
checkNamesValidationModeDNSDeploy	Checks names. Specify Ignore , Warn or Fail to determine how Address Manager handles conditions found by this check.
checkIfMXRecordsAreIPsDNSDeploy	Checks if MX records point to an IP address rather than an A or AAAA record. Specify Ignore , Warn or Fail to determine how Address Manager handles conditions found by this check.
checkIfMXRecordsPointToCNAMEsDNSDeploy	Checks if MX records point to a CNAME record rather than an A or AAAA record. Specify Ignore , Warn or Fail to determine how Address Manager handles conditions found by this check.
checkIfNSRecordsAreIPsDNSDeploy	Checks if NS record point to an IP address rather than an A or AAAA record. Specify Ignore , Warn or Fail to determine how Address Manager handles conditions found by this check.
checkIfSRVRecordsPointToCNAMEsDNSDeploy	Checks if SRV record point to a CNAME record rather than an A or AAAA record. Specify Ignore , Warn or Fail to determine how Address Manager handles conditions found by this check.
checkForNonTerminalWildcardsDNSDeploy	Checks for wildcards in zone names that do not appear as the last segment of a zone name. Specify Ignore or Warn to determine how Address Manager handles conditions found by this check.

Breaking an xHA

Breaking an xHA pair returns each server to its original stand-alone state.

The server that held the active role remains connected to Address Manager while the server that held the passive role is disconnected and has HA-NODE2 appended to its name. Each server is re-assigned its original IP address.

Break xHA

Breaks an xHA pair and returns each server to its original stand-alone state.

Output / Response

Breaks an xHA pair.

API call:

```
void breakXHAPair( long xHAServerId, boolean breakInProteusOnly )
```

Parameter	Description
xHAServerId	The object ID of the xHA server.
breakInProteusOnly	Either <i>true</i> or <i>false</i> ; determines whether or not the xHA pair breaks in Address Manager interface only. This argument breaks the xHA pair in Address Manager even if the xHA settings are not removed on the actual servers.

xHA Failover

Under normal operation, xHA automatically fails over in the event of a hardware, network or service failure related to the Active node.

However, you can perform a manual xHA failover for maintenance or verification purposes.

Failover xHA

Performs a manual xHA failover.

Output / Response

Performs a manual xHA failover.

API call:

```
void failoverXHA( long xHAServerId )
```

Parameter	Description
xHAServerId	The object ID of the xHA server.

Address Manager Objects

The other objects managed by the Address Manager API are native Address Manager objects.

These objects are part of the Address Manager server rather than the services it manages. For more information about Address Manager object types, refer to Address Manager Object Hierarchy on page 22, and to the *Address Manager Administration Guide*.

Configurations

Address Manager provides a separation between the logical design of a network and its implementation on the actual network hardware.

An administrator designs a network as a configuration. The configuration uses global elements such as users and groups, and local elements such as DNS and IP designs. When combined, these create a complete logical network design. During this process or afterward, servers (defined for each configuration) can be associated with different parts of the configuration using the various deployment roles available within the configuration.

Add Configuration

A generic method for adding configurations, DNS zones, and DNS resource records.

Output / Response

Returns the object ID for the new configuration.

API call:

```
long addEntity( long parentId, APIEntity entity )
```

Parameter	Description
parentId	For configurations, always set the parentId value to 0 (zero) , which is the root element.
entity	The configuration object, including its name, sharedNetwork, and user-defined fields.

Update Configuration

A configuration's **name** and **sharedNetwork** properties can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

Configuration Generic Methods

Configurations use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Get Configuration Setting

A method to get the configuration setting.

Output / Response

Returns the properties of the setting of the configuration in the following format:

```
attribute1=value1|attribute2=value2
```

➔ **Note:** Currently, the `getConfigurationSetting` method only returns the `OPTION_INHERITANCE` setting.

API call:

`String getConfigurationSetting(long configurationId, String settingName)`

Parameter	Description
configurationId	The object ID of the configuration in which the setting is to be located.
settingName	The name of the specific setting to be read. Only the <i>option inheritance</i> (<code>OPTION_INHERITANCE</code>) setting is supported.

Update Configuration Setting

A method to update the configuration setting.

Output / Response

Updates the configuration setting.

API call:

`void updateConfigurationSetting(long configurationId, String settingName, String properties)`

Parameter	Description
configurationId	The object ID of the configuration in which the setting is to be located.
settingName	The name of the specific setting to be read. Only the <i>option inheritance</i> (<code>OPTION_INHERITANCE</code>) setting is supported.
properties	The new properties of the configuration setting to be updated. Only the <i>disable DNS option inheritance</i> (<code>disableDnsOptionInheritance</code>) property is supported. If set to <i>true</i> , DNS options that have been configured on a zone are not inherited by the child zone. In the reverse space, DNS options that have been configured on a block are not inherited by the child block or network. ➔ Note:

Parameter	Description
	<ul style="list-style-type: none"> Disabling DNS option inheritance only affect options attached to a zone or block and the inheritance of those options by child zones, blocks, or networks. You cannot disable the inheritance of DNS options that are attached to a configuration, view, or server. These DNS options will continue to be inherited by all zones, blocks, or networks found under the object.

Groups and Users

Address Manager is designed to accommodate environments that require the ability to host multiple concurrent users who could be located in different regions. Address Manager can also be run by a single administrator.

Add Group

Adds user groups.

Output / Response

Returns the object ID for the new Address Manager user group.



Tip: To add users to a user group, use the `linkEntities()` method, specifying the user ID and the group ID. It does not matter in which order you specify the user ID and the group ID. Either of the following will add a user to a user group:

```
void linkEntities ( long user_id, long group_id, String properties )
or
void linkEntities ( long group_id, long user_id, String properties )
```

API Call:

```
long addUserGroup( String name, String properties )
```

Parameter	Description
name	The name of the user group.
properties	Adds object properties, including user-defined fields.

Update Group

A user group's **name** property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

Group Generic Methods

User groups use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Add User

Adds Address Manager users.

Output / Response

Returns the object ID for the new Address Manager user.




Tip: To add users to a user group, use the `linkEntities()` method, specifying the user ID and the group ID. It does not matter in which order you specify the user ID and the group ID. Either of the following will add a user to a user group:

```
void linkEntities ( long user_id, long group_id, String properties )
or
void linkEntities ( long group_id, long user_id, String properties )
```

API Call:

```
long addUser( String username, String password, String properties )
```

Parameter	Description
username	The name of the user.
password	The Address Manager password for the user. The password must be set even if the authenticator property option is defined.
properties	<p>A string containing user-defined fields and options listed in List of options. Multiple property values can be separated by a (pipe) character. For example: my \$properties = "email=\$email phoneNumber=\$tel authenticator=1368969 userAccessType=\$accessType"</p> <p> Note: You must add a (pipe) character at the end in the properties string.</p>

List of options

authenticator	The object ID of the external authenticator defined in Address Manager.
securityPrivilege	A security privilege type for Non-Administrator users with GUI, API, or GUI and API access. NO ACCESS is the default value.
historyPrivilege	A history privilege type for Non-Administrator users with GUI, or GUI and API access. HIDE is the default value.
email	The email address for the user. This is required.
phoneNumber	The phone number for the user.
UserType	ADMIN or REGULAR (non-administrator— REGULAR is the default value).
UserAccessType	API , GUI , or GUI and API . This is required. This string must be one of the constants listed in User Access Type on page 215.

User types and access types

UserType	UserAccessType	Privileges
ADMIN	n/a	History and Security privileges are set automatically.
REGULAR	GUI	History and Security privileges are set to a user-specific value.
REGULAR	API	Security privilege is set to a user-specific value.
REGULAR	GUI and API	History and Security privileges are set to a user-specific value.

Update User

A Address Manager user's **securityPrivilege** and **historyPrivilege** properties can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

Update User Password

Updates an Address Manager user password. You must be an Address Manager administrator to invoke this method.

Output / Response

None.

API Call:

```
void updateUserPassword( long userId, String newPassword, String[] options )
```

Parameter	Description
userId	The <code>userId</code> of an application user who is either a primary or a secondary authenticator whose password is to be updated.
newPassword	The new password for the user.
options	Reserved for future use.

User Generic Methods

Address Manager user objects use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Authenticators

Address Manager includes a fully featured authentication subsystem. The Address Manager administrator uses this system to securely log in to Address Manager and administer the system when it is being configured.

Address Manager also supports mixed-mode authentication through RADIUS, LDAP, Microsoft Active Directory, or Kerberos. Support for RSA Secure ID is accomplished through the RADIUS authentication module.

The necessary settings must be in place before Address Manager can pass authentication information to these remote systems. Also, the authentication method must be associated with a Address Manager user. This is accomplished by creating an authenticator and assigning it to a user. Authenticators are system objects that represent a connection to an external authentication system. The use of that system's native safeguards applies for communications between it and Address Manager. Address Manager acts as a proxy client for the authentication system, validating the identity of a Address Manager user without managing or validating the user's password or credentials.

After the users are authenticated against the external system, they are considered to be validated in Address Manager until they close their sessions, or until it is invalidated by a session time-out. Authentication is not a substitute for Address Manager user management. Being a Address Manager user is still a requirement to log in to the system. Authenticators move the responsibility of validating credentials to another system.

Many organizations centralize control over internal digital identities. In such scenarios, suspending or revoking credentials and password management are tightly controlled. Address Manager is designed to be deployed within all major network authentication frameworks. This lets Address Manager assist with enforcing network standards, rather than requiring a circumvention.

A user may be assigned several authenticators. These are used in order of primary-secondary.

Update Authenticator

An authenticator's **name** property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

Authenticator Generic Methods

Authenticators use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Access Rights

Address Manager is arranged as a hierarchy of objects with the server itself at the highest level. This offers a security and privilege system that is both simple and adaptive because different subsections of the server do not have separate systems.

An object may be an entire configuration, a single subnet, a tag, etc. However, this means that existing security schemes must be mapped to the Address Manager architecture. This section outlines the concepts necessary to perform this mapping.

The Address Manager server is a hierarchical structure with a configuration, user, group, or Object Tag group as the root element of the tree. Because everything in this hierarchy is an object, a user can have a different set of rights for each object within the system. However, permissions are more likely granted for a certain level as well as for everything below that level (with certain exceptions).

Access rights within Address Manager can be assigned to both users and groups. Furthermore, multiple rights can exist for the same object. Three simple rules dictate a user's access rights for any object:

- Administrators always have full control over any system object.
- Local rights take precedence over rights assigned higher in the object hierarchy.
- In the case of conflicting object rights, the most permissive right always takes precedence.

These three rules cover all of the possible cases for access rights. For more information about Address Manager user rights and security, refer to the Address Manager Administration Guide.

Add Access Right

Adds access rights to a specified object.


Output / Response

Returns the object ID for the new access right.

API Call:


```
long addAccessRight( long entityId, long userId, String value, String overrides, String properties )
```

Parameter	Description
entityId	The object ID of the entity to which the access right is being added. Set this to 0 if the access right is being added to the root level (default access rights).
userId	The object ID of the user to whom this access right applies.
value	The value of the access right being added. Valid values for this parameter are listed in Access Right Values on page 189.

Parameter	Description
overrides	A list of type-specific overrides in the following format: "objectType=accessValue objectType=accessValue"
properties	<p>A string including the following options:</p> <ul style="list-style-type: none"> • workflowLevel—valid values for this option are as follows: <ul style="list-style-type: none"> • None—changes made by the user or group take effect immediately. • Recommend—changes made by the user or group are saved as change requests and must be reviewed and approved before they take effect. • Approve—changes made by the user or group take effect immediately and the user or group can approve change requests from other users or groups. • deploymentAllowed—either true or false; to indicate whether or not the user or group can perform a full deployment of data from the configuration to a managed server. • quickDeploymentAllowed—either true or false; to indicate whether or not the user or group can instantly deploy changed DNS resource records. <p> Note:</p> <ul style="list-style-type: none"> • All these Properties are optional. • The deploymentAllowedproperty is applicable only for configuration, server or root with <i>Full</i> access. • The workflowLevelproperty is applicable only for <i>Change</i>, <i>Add</i>, or <i>Full</i> access rights.

Get Access Right

Retrieves an access right for a specified object.

-  **Note:** If the full access right is set on the parent object, the `getAccessRight()` method for the child object will retrieve the full access right even if there is a *hide* override set for the child object type. It is the caller's responsibility to evaluate the returned `APIAccessRight`'s value and overrides to determine the effective access level for the child object.

Output / Response

Returns the access right for the specified object.

API Call:

```
APIAccessRight getAccessRight(long entityId, long userId)
```

Parameter	Description
entityId	The object ID of the entity to which the access right is assigned.
userId	The object ID of the user to whom the access right is applied.

Get Access Rights for Entity

Returns an array of access rights for entities.

Output / Response

Returns an array of access right objects.

API Call:

```
APIAccessRight[] getAccessRightsForEntity( long entityId,int start, int count )
```

Parameter	Description
entityId	The object ID of the entity whose access rights are returned.
start	Indicates where in the list of child access right objects to start returning objects. The list begins at an index of 0.
count	The maximum number of access right child objects to return.

Get Access Rights for User

Returns an array of access rights for a specified user.

Output / Response

Returns an array of access right objects.

API Call:

```
APIAccessRight[] getAccessRightsForUser( long userId,int start, int count )
```

Parameter	Description
entityId	The object ID of the user whose access rights are returned.
start	Indicates where in the list of child access right objects to start returning objects. The list begins at an index of 0.
count	The maximum number of access right child objects to return.

Update Access Rights

Updates access rights for a specified object.


Output / Response

None.

API Call:

```
void updateAccessRight( long entityId, long userId, String value,String overrides, String properties )
```

Parameter	Description
entityId	The object ID of the entity to which the access right is assigned.
userId	The object ID of the user to whom the access right is assigned. This value is not mutable.
value	The new value for the access right. Valid entries are listed in Access Right Values on page 189.
overrides	A list of potentially modified type-specific overrides in the following format: "objectType=accessValue objectType=accessValue"
properties	A string including the following options: <ul style="list-style-type: none"> workflowLevel—valid values for this option are as follows: <ul style="list-style-type: none"> None—changes made by the user or group take effect immediately.

Parameter	Description
	<ul style="list-style-type: none"> • Recommend—changes made by the user or group are saved as change requests and must be reviewed and approved before they take effect. • Approve—changes made by the user or group take effect immediately and the user or group can approve change requests from other users or groups. • deploymentAllowed—either <i>true</i> or <i>false</i>; to indicate whether or not the user or group can perform a full deployment of data from the configuration to a managed server • quickDeploymentAllowed—either <i>true</i> or <i>false</i>; to indicate whether or not the user or group can instantly deploy changed DNS resource records. <p> Note:</p> <ul style="list-style-type: none"> • All these Properties are <i>optional</i>. • The deploymentAllowed property is applicable only for configuration, server or root with <i>Full</i> access. • The workflowLevel property is applicable only for <i>Change</i>, <i>Add</i>, or <i>Full</i> access rights.

Delete Access Rights

Deletes an access right for a specified object.

Output / Response

None.

API Call:

```
void deleteAccessRight( long entityId, long userId )
```

Parameter	Description
entityId	The object ID of the entity to which the access right is assigned.
userId	The object ID of the user to whom this access right is applied.

Object Tag Groups

Object tags can change the entire scheme by which users navigate Address Manager. By tagging various objects, companies can assign privileges based on existing business authority regimes, and limit access to system objects using familiar business models.

Address Manager object tags are arranged in a hierarchical tree structure. This should accommodate most element-based XML designs, because any realistic number of elements are supported at each level of the hierarchy below a top-level or root tag known as a tag group. The system supports more than one hundred levels of tags, so it can accommodate complex nested structures.

The object tagging structure comprises large sets of XML elements that are without attributes. They begin with a root element and all subsequent tags belong to branches below the tag group in a series of parent-child relationships. The tag groups cannot be applied to objects.

Add Object Tag Group

Adds object tag groups.

Output / Response

Returns the object ID for the new tag group.

API Call:

```
long addTagGroup( String name, String properties )
```

Parameter	Description
name	The name of the tag group.
properties	Adds object properties, including user-defined fields.

Update Object Tag Group

A tag group's **name** property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

Object Tag Group Generic Methods

This object implements the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Object Tags

Object tag groups use the generic `get()` and `delete()` methods for entities.

Add Object Tag

Adds object tags.

Output / Response

Returns the object ID for the new object tag.

API Call:

```
long addTag( long parentid, String name, String properties )
```

Parameter	Description
parentid	The object ID of the parent for this object tag. The parent is either an object tag or an object tag group.
name	The name of the object tag.
properties	Adds object properties, including user-defined fields.

Assign Object Tag

Assigns object tags to objects through the Address Manager API.



Note: This method is deprecated. Using this method now returns an error message. Use the `linkEntities()` method instead. For more information, see [Get Linked Entities](#) on page 53.

Output / Response

None.

API Call:

```
void tagEntity( long entityId, String tagId )
```

Parameter	Description
entityId	The object ID of the entity to which the tag is assigned.
tagId	The object ID of the tag that is assigned.

Remove Object Tag

Removes object tags from specified objects.



Note: This method is deprecated. Using this method now returns an error message. Use the `unlinkEntities()` method instead. For more information, refer to [Unlink Entities](#) on page 54.

Output / Response

None.

API Call:

```
void tagEntity(long entityId, String tagId)
```

Parameter	Description
entityId	The object ID of the entity from which the tag is to be removed.
tagId	The object ID of the tag to be removed.

Update Object Tag

An object tag's **name** property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

Object Tag Generic Methods

Object tags use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Locations

The Location feature in Address Manager helps organizations with a large global network infrastructure manage their network objects and standardize location information.

You can use the Location feature to customize your network location structure by adding your own location objects under default master location objects. Master location objects are available by default in Address Manager. You cannot create, edit or delete any default master location objects.

Add a Location

Adds a location object under a parent location object.

`addEntity()` is a generic method for adding Address Manager API objects.

Output / Response

Returns the object ID for the newly added location object in Address Manager.

API Call:

```
long addEntity(long parentId, APIEntity entity)
```

Parameter	Description
parentId	The object ID of the parent location object.

Parameter	Description
entity	The location object. For a list of supported properties, refer to Location on page 246.

Get Location By Code


Use this method to get the location object with the specified hierarchical location code.

Output / Response

Returns the entity that matches the specified hierarchical location code. If no entity is found, returns an *empty APIEntity*. For a list of supported properties, refer to [Location](#) on page 246.

API Call:

```
APIEntity getLocationByCode( String code )
```

Parameter	Description
code	<p>The hierarchical location code consists of a set of 1 to 3 alpha-numeric strings separated by a space. The first two characters indicate a country, followed by next three characters which indicate a city in UN/LOCODE. New custom locations created under a UN/LOCODE city are appended to the end of the hierarchy. For example, CA TOR OF1 indicates:</p> <ul style="list-style-type: none"> • CA—Canada • TOR—Toronto • OF1—Office 1 <p> Note: The code is case-sensitive. It must be all UPPER CASE letters. The county code and child location code should be alphanumeric strings.</p>

Get All Used Locations

Use this method to get all location objects that are used to annotate other objects.

Output / Response

Returns an array of location APIEntity objects. For a list of supported properties, refer to [Location](#) on page 246.

API Call:

```
APIEntity[] getAllUsedLocations()
```

Database Management

BlueCat Address Manager comes with built-in PostgreSQL database to store data such as DHCP and DNS objects, configuration information, notifications, server information, events, and alerts.

The Address Manager Database functions transparently to store the data. Because the database is built inside Address Manager, you do not need to install or set up the database separately to get it functioning. However, you need to configure administrative features such as database backup, replication, restoration, or disaster recovery from the Address Manager Administration Console.

Configure Replication

Enables database replication on a remote system in order to automate the setup of replication between two or three Address Manager systems. This API method must be run against the Address Manager system that will be primary.

Output / Response

None.

API Call:

```
void configureReplication( String standbyServer, boolean compressReplication, long replicationQueueThreshold, long replicationBreakThreshold, String properties )
```

Parameter	Description
standbyServer	The IP address of the standby server. ➔ Note: The standby server must be accessible from the primary server and must have database access from the primary server. To enable database access, refer to the Configuring database replication section in the <i>Address Manager Administration Guide</i> .
compressReplication	The boolean value. Set to true to compress the database replication files. ➔ Note: Compressing database replication files is a resource-intensive process that might affect system performance. Use caution when performing this action.
replicationQueueThreshold	A value to specify the threshold size of the replication directory in megabytes (MB). The valid values are in the range of 16 to 99999999.
replicationBreakThreshold	A value to specify the threshold size of the replication break in gigabytes (GB). The valid values are in the range of 5 to 30. This value multiplied by 1024 must be greater than the value of <i>replicationQueueThreshold</i> .
properties	A string containing the following property: <ul style="list-style-type: none">• secondStandbyServer—the IP address of the second standby server. This is optional. ➔ Note: Any property string other than <i>secondStandbyServer</i> option will be ignored.

Purge History

Run the purge function.

Output / Response

Returns **0** (zero) when the purge service has successfully completed.

The following codes will be returned if you specify the *waitOption* parameter to *true*. These returning codes will be logged in the `/var/log/jetty/server.log` file:

- **33** – <untilWhenTimestamp> is either empty or does not contain a hyphen and no valid value found in either of the two numeric alternatives.
- **35** – <untilWhenTimestamp> contains a hyphen but is not a valid timestamp.

The details of the output or the reason for failure will be logged in `/tmp/purge_results.out`.

API Call:

```
int purgeHistoryNow( String untilWhenTimestamp, int numberOfDaysToKeep, int
numberOfMonthsToKeep, boolean waitOption )
```

- ➔ **Note:** This API method will examine each of the following three retention-period specifying parameters in sequential order to decide which parameter value it will use to run the purge service. The first valid parameter will be used:

1. *untilWhenTimestamp*
2. *numberOfDaysToKeep*
3. *numberOfMonthsToKeep*

Parameter	Description
untilWhenTimestamp	The string specifying the point in time after which history is to be preserved. The valid timestamp pattern is <code>YYYY-MM[-DD[HH:MM:SS[.mmm]]]</code> . Set this parameter with an empty string ("") if you are using either <i>numberOfDaysToKeep</i> or <i>numberOfMonthsToKeep</i> . Using this parameter will not archive any data.
numberOfDaysToKeep	The number of days for which the data will be preserved in the database. The valid value for this parameter is between 1 and 3650. Set this parameter to -1 if you are using either <i>untilWhenTimestamp</i> or <i>numberOfMonthsToKeep</i> . Using this parameter will archive all history that will be purged and overwrite existing archive files in the <code>/data/Archive</code> directory.
numberOfMonthsToKeep	The number of months for which the data will be preserved in the database. The valid value for this parameter is between 0 to infinite months. Set this parameter to -1 if you are using either <i>untilWhenTimestamp</i> or <i>numberOfDaysToKeep</i> . Using this parameter will not archive any data.
waitOption	The Boolean value. If set to <i>true</i> , the purge will be performed and the result will be returned when completed. The default value is <i>false</i> .

Devices

A device is an actual physical component, such as a router or printer or other equipment to which one or more IP addresses are assigned.

Devices are organized by device types and device sub-types. A device type is a general category of devices; a device sub-type is a more specific category of devices. For example, a general device type might be Printers. More specific device sub-types might include Laser Printers, Plotters, and Imagesetters.

Add Device

Adds a device to a configuration.

Output / Response

Returns the object ID of the new device.

API Call:

```
long addDevice( long configurationId, String name, long deviceId, long
deviceSubtypeId, String ip4Addresses, String ip6Addresses, String properties )
```

Parameter	Description
configurationId	The object ID of the configuration in which the device is to be located.
name	The descriptive name of the device.

Parameter	Description
deviceTypeId	The object ID of the device type with which the device is associated. The value can be 0 if you do not wish to associate a device type to the device you are adding.
deviceSubTypeId	The object ID of the device sub-type with which the device is associated. The value can be 0 if you do not wish to associate a device sub-type to the device you are adding.
ip4Addresses	One or more IPv4 addresses to which the device is assigned. Specify multiple addresses in a comma-delimited list.
ip6Addresses	One or more IPv6 addresses to which the device is assigned. Specify multiple addresses in a comma-delimited list.
properties	Adds object properties, including user-defined fields.

Add Device Type

Adds a device type to Address Manager. Use device types and device sub-types to categorize and organize devices on the network.

Output / Response

Returns the object ID of the new device type

API Call:

```
long addDeviceType( String name, String properties )
```

Parameter	Description
name	The descriptive name for the device type.
properties	Adds object properties, including user-defined fields.

Add Device Subtype

Adds a device sub-type to Address Manager. Use device types and device sub-types to categorize and organize devices on the network.

Output / Response

Returns the object ID of the new device sub-type

API Call:

```
long addDeviceSubtype( long parentId, String name, String properties )
```

Parameter	Description
parentId	The object ID of the parent device type object.
name	The descriptive name for the device sub-type.
properties	Adds object properties, including user-defined fields.

MAC Pools

Media Access Control (MAC) pools are used to group MAC addresses for functionality such as Network Access Control (NAC).

Each MAC pool can be linked to multiple MAC addresses, and each MAC address can be linked to multiple IP addresses of different networks. However, each MAC address can belong to only one MAC pool, and each IP address can belong to only one MAC address. The MAC pools include one default global 'Deny' pool object that the user cannot delete. All pools created by the user can be deleted. A MAC pool contains a name (required), and optional links to MAC addresses.

Get MAC Addresses in Pool

Returns a list of the MAC address objects within a specified MAC pool.

➔ **Note:** This method is deprecated. Using this method now returns an error message. Use the `getLinkedEntities()` method instead. For more information, see [Get Linked Entities](#) on page 53.

Output / Response

Returns an array of MAC address objects.

API Call:

```
APIEntity[] getMACAddressesInPool( long macPoolId, int start, int count )
```

Parameter	Description
macPoolId	The object ID for the MAC pool.
start	Indicates where in the list of children to start returning objects. The list begins at an index of 0.
count	This is the maximum number of child objects to return.

Add MAC Pool

Returns the object ID for the new MAC pool.

You can use the generic `addEntity()` method to add a MAC pool.

Output / Response

Returns the object ID for the new MAC pool.

API Call:

```
long addEntity( long parentID, APIEntity entity )
```

Parameter	Description
parentId	The object ID of the parent Configuration to which the MAC pool is added.
entity	The MAC pool object with its name defined.

Update MAC Pool

A MAC pool's **name** property can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

MAC Pool Generic Methods

MAC pools use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

MAC Addresses

MAC address objects are used to reference the MAC addresses of endpoints.

Add MAC Address

Adds MAC addresses.

➔ **Note:** To assign a MAC address to the DENY MAC pool, use the `denyMACAddress()` method. For more information, see [Deny MAC Address](#) on page 181.

Output / Response

Returns the object ID for the new MAC address.

API Call:

```
long addMACAddress( long configurationId, String macAddress, String properties )
```

Parameter	Description
configurationId	The object ID of the parent configuration in which the MAC address resides.
macAddress	The MAC address in the format nnnnnnnnnnnn, nn-nn-nn-nn-nn-nn or nn:nn:nn:nn:nn:nn, where nn is a hexadecimal value.
properties	Adds object properties, including the name, MAC Pool ID (macPool), and user-defined fields.

Associate MAC Address

Associates a MAC address with a MAC pool.

⚠ **Attention:** To assign a MAC address to the DENY MAC pool, use the `denyMACAddress()` method. For more information, see [Deny MAC Address](#) on page 181.

Output / Response

None.

API Call:

```
void associateMACAddressWithPool( long configurationId, String macAddress, long poolId )
```

Parameter	Description
configurationId	The object ID of the parent configuration in which the MAC address resides.
macAddress	The MAC address in the format nnnnnnnnnnnn, nn-nn-nn-nn-nn-nn or nn:nn:nn:nn:nn:nn, where nn is a hexadecimal value.
poolId	The object ID of the MAC pool with which this MAC address is associated.

Deny MAC Address

Denies MAC addresses.

Output / Response

None.

API Call:

```
void denyMACAddress( long configurationId, String macAddress )
```

Parameter	Description
configurationId	The object ID of the parent configuration in which the MAC address resides.
macAddress	The MAC address in the format nnnnnnnnnnnn, nn-nn-nn-nn-nn or nn:nn:nn:nn:nn:nn, where nn is a hexadecimal value.

Is Address Allocated?

Queries a MAC address to determine if the address has been allocated to an IP address.

Output / Response

Returns a Boolean value indicating whether the address is allocated.

API Call:

```
boolean isAddressAllocated( long configurationId, String ipAddress, String macAddress )
```

Parameter	Description
configurationId	The object ID of the parent configuration in which the MAC address resides.
ipAddress	The IPv4 DHCP allocated address to be checked against the MAC address.
macAddress	The MAC address in the format nnnnnnnnnnnn, nn-nn-nn-nn-nn or nn:nn:nn:nn:nn:nn, where nn is a hexadecimal value.

Update MAC Address

A MAC address's **name** and **macPoolId** properties can be updated using the generic `update()` method.

For more information, see [Updating Objects](#) on page 50.

MAC Address Generic Methods

MAC addresses use the generic `get()` and `delete()` methods for entities.

For more information, see [Getting Objects](#) on page 44 and [Deleting Objects](#) on page 52.

Workflow Change Requests

You can use change requests to manage the creation of network, resource records, zones, and IP address assignments. Workflow permissions are assigned to users along with access rights.

Users with a default access right of **Change**, **Add**, or **Full Access** can be assigned one of these workflow levels:

- **None**—the user is not affected by the change request process and can create networks, resource records, zones, and IP address assignments. Users with the **None** level cannot access or work with workflow change requests.
- **Recommend**—when the user creates, edits, or deletes a network, resource record, zone, or IP address assignment, Address Manager creates a change request for the object. The change request must be approved before the object is actually created, edited, or deleted. Users with the **Recommend** level can review their change requests.
- **Approve**—the user can approve change requests made by other users. Users with the **Approve** level can create, edit, or delete networks, resource records, zones, and IP address assignments.

For information about adding access rights and Workflow Levels, refer to [Add Access Right](#) on page 195. Users who are assigned the workflow level of **Recommend** create a change request each time they add, edit, or delete a network, resource record, zone, or IP address.

The following objects support workflow mode operations:

- Zone
- HostRecord
- AliasRecord (CName)
- MXRecord
- TXTRecord
- GenericRecord
- HINFORecord
- NAPTRRecord
- SRVRecord
- IP4Network
- IP4Address

The following API operations support workflow mode operations:

- Add (for all objects except IP4Address)
- Update
- Delete

Migration

You can use XML files to migrate data from other systems into Address Manager. For more information on the migration document type definition (DTD) and performing migrations from the Address Manager web interface, see **Migration** in the *Address Manager Administration Guide* and Address Manager online help.

The Address Manager API provides two methods for managing the migration service:

- `migrateFile()` migrates a specified XML file in to Address Manager.
- `isMigrationRunning()` indicates if migrations are queued or in progress.

Migrate a File

Migrates the specified XML file in to Address Manager. The file must be located in the **/data/migration/incoming** directory on the Address Manager server. The file name must not include a path.

Output / Response

None.

API Call:

```
void migrateFile( String filename )
```

Parameter	Description
filename	The filename of the XML file in the data/migration/incoming directory. Do not include a path in the filename.

Migration Status

Returns true or false to indicate if the migration service is running. Specify a filename to determine if the specified file is migrating. Specify an empty string ("") to determine if any migration files are migrating or queued for migration.

API Call:

```
boolean isMigrationRunning( String filename )
```

Parameter	Description
filename	The filename of the XML file in the data/migration/incoming directory. Do not include a path in the filename. This value can be empty.

Output / Response

Returns a Boolean value indicating if the specified file is currently migrating. When an *empty string* ("") is specified for the filename, returns a true if there are any migration files queued for migration or currently migrating.

Collecting Data

The Address Manager API provides three methods for gathering data from the Address Manager database.

Start Probe

Starts collecting data from Address Manager's database using pre-defined SQL queries.

Output / Response

None.

API Call:

```
void startProbe( String definedProbe, String properties )
```

Parameter	Description
definedProbe	Pre-defined SQL queries that will be triggered to collect data. The available values are <i>LEASE_COUNT_PER_DATE</i> and <i>NETWORK_BLOOM</i> .
properties	Reserved for future use.

Get Probe Status

Returns the status of the triggered data collection process.

Output / Response

This method will return a pre-defined value from 0 to 3, depending on the status of the data collection process. For more information, refer to [Probe Status Values](#) on page 217.

API Call:

```
int getProbeStatus( String definedProbe )
```

Parameter	Description
definedProbe	Pre-defined SQL queries that have been triggered to collect data. The available values are <i>LEASE_COUNT_PER_DATE</i> and <i>NETWORK_BLOOM</i> .

Get Probe Data

Returns the JSON response from the properties field of the APIData object.

Output / Response

Returns the JSON response from the properties field of the APIData object.

API Call:

APIData getProbeData(String *definedProbe*, String *properties*)

Parameter	Description
definedProbe	Pre-defined SQL queries that will be triggered to collect data. The available values are <i>LEASE_COUNT_PER_DATE</i> and <i>NETWORK_BLOOM</i> .
properties	Reserved for future use.

API Constants

Topics:

- [Access Right Values](#)
- [Additional IP Service Type](#)
- [Configuration Setting](#)
- [Deployment Services](#)
- [Deployment Status](#)
- [Device Properties](#)
- [DHCP Class Match Criteria](#)
- [DHCP Client Options](#)
- [DHCP6 Client Options](#)
- [DHCP Custom Option Types](#)
- [DHCP Deployment Role Types](#)
- [DHCP Service Options](#)
- [DHCPServiceOptionConstants](#)
- [DHCP6 Service Options](#)
- [DNS Deployment Role Type](#)
- [DNS Options](#)
- [DNS Option Values](#)
- [DNSSEC Key Format](#)
- [DNS Zones Deployment Validation Check](#)
- [Entity Categories](#)
- [ENUM Services](#)
- [IP Assignment Action Values](#)
- [IP Discovery Type](#)
- [Object Properties](#)
- [Object Types](#)
- [Option Types](#)
- [PositionRangeBy](#)
- [Response Policy Type](#)
- [Response Policy Search Scopes](#)
- [Reverse Zone Format Type](#)
- [Server Capability Profiles](#)
- [Service Types](#)
- [SNMP Version](#)
- [SNMP Security Levels](#)
- [SNMP Authentication Type](#)
- [SNMP Privacy Type](#)
- [Traversal Methodology](#)
- [User Access Type](#)

The Address Manager API uses various constants in its methods. To accommodate changes in values, these constants are wrapped in various libraries (such as Java and Perl).

The proceeding tables describe the Library (type of constant), **Property Key** which is the name of the property in the library, and **Property Value** which is the value to be used for the actual API call. When you are not using the provided wrappers for Java and Perl, you should use the value in the **Property Value** field.

- *User-defined Field Type*
- *User-defined Field Validator Properties*
- *User History Privileges*
- *User Security Privileges*
- *User Type*
- *Vendor Profile Option Types*
- *Workflow Levels*
- *Defined Probe Values*
- *Probe Status Values*

Access Right Values

Constants used in Access Right API methods.

Property Key	Property Value
HideAccess	HIDE
ViewAccess	VIEW
AddAccess	ADD
ChangeAccess	CHANGE
FullAccess	FULL

Additional IP Service Type

Constants used in the Additional IP Service Type.

Property Key	Property Value
SERVICE	service
LOOPBACK	loopback

Configuration Setting

Constants used in the Configuration Setting.

Property Key	Property Value
OPTION_INHERITANCE	OPTION_INHERITANCE

Deployment Services

Constants used in Deployment Services API methods.

Property Key	Property Value
DNS	DNS
DHCP	DHCP
TFTP	TFTP
DHCPv6	DHCPv6

Deployment Status

Constants used in the Deployment Status API method.

Property Key	Property Value
EXECUTING	-1
INITIALIZING	0

Property Key	Property Value
QUEUED	1
CANCELLED	2
FAILED	3
NOT_DEPLOYED	4
WARNING	5
INVALID	6
DONE	7
NO_RECENT_DEPLOYMENT	8

Device Properties

Constants used in the Device Properties.

Property Key	Property Value
DEVICE_ROUTER	ROUTER
DEVICE_SWITCH	L2SWITCH
DEVICE_INTERFACE	INTERFACE
DEVICE_VLAN	VLAN
DEVICE_LOCATION	location
DEVICE_IP	ip
TOTAL_INTERFACES	totalInterfaces
TOTAL_VLANS	totalVlans
TOTAL_ARPS	totalArps
TOTAL_MACS	totalMacs
TOTAL_NETWORKS	totalNetworks
TOTAL_IPS	totalIps
INTERFACE_ALIAS	alias
INTERFACE_DESC	description
INTERFACE_INDEX	ifIndex
INTERFACE_SPEED	speed
INTERFACE_CONNECTOR	connector
MAC_ADDRESS	physicalAddress
VLAN_ID	vlanId
PORT_MODE	portMode

DHCP Class Match Criteria

Constants used in the DHCP Class Match Criteria API method.

Property Key	Property Value
DHCP_CLASS_HARDWARE	MATCH_HARDWARE
DHCP_CLASS_CLIENT_ID	MATCH_DHCP_CLIENT_ID
DHCP_CLASS_VENDOR_ID	MATCH_DHCP_VENDOR_ID
DHCP_CLASS_AGENT_CIRCUIT_ID	MATCH_AGENT_CIRCUIT_ID
DHCP_CLASS_AGENT_REMOTE_ID	MATCH_AGENT_REMOTE_ID
DHCP_CLASS_CUSTOM_MATCH	CUSTOM_MATCH
DHCP_CLASS_CUSTOM_MATCH_IF	CUSTOM_MATCH_IF

DHCP Client Options

Constants used in the DHCP Client Options API method.

Property Key	Property Value
TIME_OFFSET	time-offset
ROUTER	router
TIME_SERVER	time-server
IEN_NAME_SERVER	ien-name-server
DNS_SERVER	dns-server
LOG_SERVER	log-server
COOKIE_SERVER	cookie-server
LPR_SERVER	lpr-server
IMPRESS_SERVER	impress-server
RESOURCE_LOCATION_SERVER	resource-location-server
HOST_NAME	host-name
BOOT_SIZE	boot-size
MERIT_DUMP_FILE	merit-dump-file
DOMAIN_NAME	domain-name
SWAP_SERVER	swap-server
ROOT_PATH	root-path
EXTENSIONS_PATH	extensions-path
IP_FORWARDING	ip-forwarding
NON_LOCAL_SOURCE_ROUTING	non-local-source-routing
POLICY_FILTER_MASKS	policy-filter-masks

Property Key	Property Value
MAX_DATAGRAM_REASSEMBLU	max-datagram-reassembly
DEFAULT_IP_TTL	default-ip-ttl
PATH_MTU_AGING_TIMEOUT	path-mtu-aging-timeout
PATH_MTU_PLATEAU_TABLE	path-mtu-plateau-table
INTERFACE_MTU	interface-mtu
ALL_SUBNETS_LOCAL	all-subnets-local
BROADCAST_ADDRESS	broadcast-address
PERFORM_MASK_DISCOVERY	perform-mask-discovery
MASK_SUPPLIER	mask-supplier
ROUTER_DISCOVERY	router-discovery
ROUTER_SOLICITATION_ADDRESS	router-solicitation-address
STATIC_ROUTES	static-routes
TRAILER_ENCAPSULATION	trailer-encapsulation
ARP_CACHE_TIMEOUT	arp-cache-timeout
IEEE_802_3_ENCAPSULATION	ieee-802-3-encapsulation
DEFAULT_TCP_TTL	default-tcp-ttl
TCP_KEEP_ALIVE_INTERVAL	tcp-keep-alive-interval
TCP_KEEP_ALIVE_GARBAGE	tcp-keep-alive-garbage
NIS_DOMAIN	nis-domain
NIS_SERVER	nis-server
VENDOR_ENCAPSULATED_OPTIONS	vendor-encapsulated-options
NTP_SERVER	ntp-server
WINS_NBNS_SERVER	wins-nbns-server
NETBIOS_OVER_TCP_IP_NBDD	netbios-over-tcp-ip-nbdd
WINS_NBT_NODE_TYPE	wins-nbt-node-type
NETBIOS_SCOPE_ID	netbios-scope-id
X_WINDOW_FONT_MANAGER	x-window-font-manager
X_WINDOW_DISPLAY_MANAGER	x-window-display-manager
NETWARE_IP_DOMAIN	nwip.domain
NETWARE_IP_NSQ_BROADCAST	nwip.nsq-broadcast
NETWARE_IP_PREFERRED_DSS	nwip.preferred-dss
NETWARE_IP_NEAREST_NWIP_SERVER	nwip.nearest-nwip-server
NETWARE_IP_AUTO_RETRIES	nwip.auto-retries
NETWARE_IP_AUTO_RETRY_DELAY	nwip.auto-retry-delay

Property Key	Property Value
NETWARE_IP_1_1_COMPATIBILITY	nwip.1-1-compatibility
NETWARE_IP_PRIMARY_DSS	nwip.primary-dss
NIS_PLUS_DOMAIN_NAME	nis-plus-domain-name
NIS_PLUS_SERVER	nis-plus-server
TFTP_SERVER_NAME	tftp-server-name
BOOT_FILE_NAME	boot-file-name
MOBILE_IP_HOME_AGENT	mobile-ip-home-agent
SMTP_SERVER	smtp-server
POP3_SERVER	pop3-server
NNTP_SERVER	nntp-server
WWW_SERVER	www-server
FINGER_SERVER	finger-server
IRC_SERVER	irc-server
STREET_TALK_SERVER	street-talk-server
STREET_TALK_DIRECTORY_ASSISTANCE_SERVER	street-talk-directory-assistance-server
SLP_DIRECTORY_AGENT	slp-directory-agent
SLP_SERVICE_SCOPE	slp-service-scope
NDS_SERVER	nds-server
NDS_TREE_NAME	nds-tree-name
NDS_CONTEXT	nds-context
UAP_SERVER	uap-server
NAME_SERVICE_SEARCH	name-service-search
DOMAIN_SEARCH	domain-search
SIP_SERVERS	sip-server
CLASSLESS_STATIC_ROUTE_OPTION	classless-static-route-option
CCC_PRIMARY_DHCP_SERVER_ADDRESS	cablelabs.primary-dhcp-server
CCC_SECONDARY_DHCP_SERVER_ADDRESS	cablelabs.secondary-dhcp-server
CCC_PROVISIONING_SERVER_ADDRESS	cablelabs.provisioning-server
CCC_AS_BACKOFF_AND_RETRY	cablelabs.as-backoff-retry
CCC_AP_BACKOFF_RETRY	cablelabs.ap-backoff-retry
CCC_KERBEROS_REALM_NAME	cablelabs.kerberos-realm-name
CCC_TICKET_GRANTING_SERVER_UTILIZATION	cablelabs.ticket-granting-server-utilization
CCC_PROVISIONING_TIMER_VALUE	cablelabs.provisioning-timer-value

Property Key	Property Value
TFTP_SERVER_ADDRESS	tftp-server
IP_TELEPHONE	ip-telephone
WPAD_URL	wpad-url

DHCP6 Client Options

Constants used in the DHCP6 Client Options API method.

Property Key	Property Value
UNICAST	unicast
DNS_SERVERS	dns-servers
DOMAIN_SEARCH_LIST	domain-search-list
SNTP_SERVERS	sntp-servers
INFORMATION_REFRESH_TIME	information-refresh-time
wpad-url	

DHCP Custom Option Types

Constants used in the DHCP Custom Option Types.

Property Key	Property Value
IP4	IP4
TEXT	TEXT
UNSIGNED_INT_8	UNSIGNED_INT_8
UNSIGNED_INT_16	UNSIGNED_INT_16
UNSIGNED_INT_32	UNSIGNED_INT_32
UNSIGNED_INT_64	UNSIGNED_INT_64
SIGNED_INT_8	SIGNED_INT_8
SIGNED_INT_16	SIGNED_INT_16
SIGNED_INT_32	SIGNED_INT_32
BOOLEAN	BOOLEAN
IP4_MASK	IP4_MASK
IP4_RANGE	IP4_RANGE
IP4_BLOCK	IP4_BLOCK
STRING	STRING
BINARY	BINARY
ENCAPSULATED	ENCAPSULATED

DHCP Deployment Role Types

Constants used in DHCP Deployment Role API methods.

Property Key	Property Value
NONE	NONE
MASTER	MASTER

DHCP Service Options

Constants used in the Deployment Status API method.

Property Key	Property Value
DEFAULT_LEASE_TIME	default-lease-time
MAX_LEASE_TIME	max-lease-time
MIN_LEASE_TIME	min-lease-time
CLIENT_UPDATES	client-updates
DDNS_DOMAINNAME	ddns-domainname
DDNS_HOSTNAME	ddns-hostname
DDNS_REV_DOMAINNAME	ddns-rev-domainname
DDNS_TTL	ddns-ttl
DDNS_UPDATES	ddns-updates
PING_CHECK	ping-check
ALWAYS_BROADCAST	always-broadcast
ALWAYS_REPLY_RFC1048	always-reply-rfc1048
DYNAMIC_BOOTP_LEASE_LENGTH	dynamic-bootp-lease-length
FILENAME	filename
GET_LEASE_HOSTNAMES	get-lease-hostnames
MIN_SECS	min-secs
NEXT_SERVER	next-server
SERVER_IDENTIFIER	server-identifier
SITE_OPTION_SPACE	site-option-space
STASH_AGENT_OPTIONS	stash-agent-options
UPDATE_OPTIMIZATION	update-optimization
UPDATE_STATIC_LEASES	update-static-leases
USE_LEASE_ADDR_FOR_DEFAULT_ROUTE	use-lease-addr-for-default-route
ONE_LEASE_PER_CLIENT	one-lease-per-client
ALLOW_MAC_POOL	allow-mac-pool

Property Key	Property Value
DENY_MAC_POOL	deny-mac-pool
DENY_UNKNOWN_MAC_ADDRESSES	deny-unknown-mac-addresses
LOAD_BALANCE_OVERRIDE	load-balance-override
LOAD_BALANCE_SPLIT	load-balance-split
MCLT	mclt
MAX_RESPONSE_DELAY	max-response-delay
MAX_UNACKED_UPDATES	max-unacked-updates
DHCP_CLASS_LEASE_LIMIT	dhcp-class-lease-limit
ALLOW_DHCP_CLASS_MEMBERS	allow-dhcp-class-members
DENY_DHCP_CLASS_MEMBERS	deny-dhcp-class-members
APPLY_MAC_AUTHENTICATION_POLICY	apply-mac-authentication-policy
DENY_DHCP_CLIENTS	deny-dhcp-clients
CONFLICT_DETECTION	conflict-detection
UPDATE_CONFLICT_DETECTION	update-conflict-detection
DO_REVERSE_UPDATES	do-reverse-updates

DHCPServiceOptionConstants

Constants used in the DHCPServiceOption API method.

Property Key	Property Value
DDNS_HOSTNAME_TYPE_IP	ip
DDNS_HOSTNAME_TYPE_MAC	mac
DDNS_HOSTNAME_TYPE_FIXED	fixed
DDNS_HOSTNAME_TYPE_DUID	duid
DDNS_HOSTNAME_POSITION_APPEND	append
DDNS_HOSTNAME_POSITION_PREPEND	prepend

DHCP6 Service Options

Constants used in the DHCP6 service options API method.

Property Key	Property Value
DEFAULT_LEASE_TIME	default-lease-time
CLIENT_UPDATES	client-updates
DDNS_DOMAINNAME	ddns-domainname
DDNS_HOSTNAME	ddns-hostname
DDNS_TTL	ddns-ttl

Property Key	Property Value
DDNS_UPDATES	ddns-updates
LIMIT_ADDRESSES_PER_IA	limit-addresses-per-ia
DO_REVERSE_UPDATES	do-reverse-updates
SERVER_PREFERENCE	server-preference
PREFERRED_LIFETIME	preferred-lifetime
RAPID_COMMIT	rapid-commit

DNS Deployment Role Type

Constants used in the reverse zone format type.

Property Key	Property Value
NONE	NONE
MASTER	MASTER
MASTER_HIDDEN	MASTER_HIDDEN
SLAVE	SLAVE
SLAVE_STEALTH	SLAVE_STEALTH
FORWARDER	FORWARDER
STUB	STUB
RECURSION	RECURSION
AD_MASTER	AD_MASTER

DNS Options

Constants used in the DNS Options.

Property Key	Property Value
ALLOW_XFER	allow-xfer
ALSO_NOTIFY	also-notify
ALLOW_DDNS	allow-ddns
ALLOW_RECURSION	allow-recursion
ALLOW_QUERY	allow-query
FORWARDING_POLICY	forwarding-policy
FORWARDING	forwarding
NOTIFY	notify
MAX_CACHE_TTL	max-cache-ttl
MAX_NEG_CACHE_TTL	max-neg-cache-ttl
TRANSFERS_IN	transfers-in

Property Key	Property Value
TRANSFERS_OUT	transfers-out
TCP_CLIENTS	tcp-clients
MAX_TRANSFER_TIME_OUT	max-transfer-time-out
MAX_TRANSFER_TIME_IN	max-transfer-time-in
MAX_TRANSFER_IDLE_OUT	max-transfer-idle-out
MAX_TRANSFER_IDLE_IN	max-transfer-idle-in
TRANSFER_FORMAT	transfer-format
MAX_CACHE_SIZE	max-cache-size
RECURSIVE_CLIENTS	recursive-clients
TRANSFERS_PER_NS	transfers-per-ns
LAME_TTL	lame-ttl
ALLOW_UPDATE_FORWARDING	allow-update-forwarding
VERSION	version
MATCH_CLIENTS	match-clients
DENY_CLIENTS	deny-clients
CACHE	cache
ALLOW_NOTIFY	allow-notify
ZONE_DEFAULT_TTL	zone-default-ttl
DNSSEC_ENABLE	dnssec-enable
DNSSEC_VALIDATION	dnssec-validation
DNSSEC_KEY_DIRECTORY	dnssec-key-directory
DNSSEC_TRUST_ANCHORS	dnssec-trust-anchors
DNSSEC_MUST_BE_SECURE	dnssec-must-be-secure
ALLOW_QUERY_CACHE	allow-query-cache
DNSSEC_ACCEPT_EXPIRED	dnssec-accept-expired
START_OF_AUTHORITY	start-of-authority
CLASSLESS_REVERSE_ZONE_FORMAT	classless-reverse-zone-format
TSIG_KEY_FOR_SERVER_PAIR	tsig-key-for-server-pair
SLAVE_ZONE_NOTIFICATIONS	slave-zone-notifications

DNS Option Values

Constants used in the DNS options values.

Property Key	Property Value
SINGLE	SINGLE

Property Key	Property Value
MANY_ANSWERS	MANY_ANSWERS
FIRST	FIRST
ONLY	ONLY

DNSSEC Key Format

Constants used in the DNSSEC Key Format.

Property Key	Property Value
TRUST_ANCHOR	TRUST_ANCHOR
DNS_KEY	DNS_KEY
DS_RECORD	DS_RECORD

DNS Zones Deployment Validation Check

Constants used in the DNS Zones Deployment Validation Check.

Property Key	Property Value
FAIL	FAIL
WARN	WARN
IGNORE	IGNORE
NONE	NONE
FULL	FULL
FULL_SIBLING	FULL_SIBLING
LOCAL	LOCAL
LOCAL_SIBLING	LOCAL_SIBLING

Entity Categories

Constants used in the Entity Categories.

Property Key	Property Value
all	ALL
admin	ADMIN
Configuration	CONFIGURATION
deploymentOptions	DEPLOYMENT_OPTIONS
deploymentRoles	DEPLOYMENT_ROLES
deploymentSchedulers	DEPLOYMENT_SCHEDULER
dhcpClassObjects	DHCPCLASSES_OBJECTS

Property Key	Property Value
dhcpNACPolicies	DHCPNACPOLICY_OBJECTS
IP4Objects	IP4_OBJECTS
IP6Objects	IP6_OBJECTS
MACPoolObjects	MACPOOL_OBJECTS
resourceRecords	RESOURCE_RECORD
servers	SERVERS
tags	TAGS
tasks	TASKS
TFTPObjects	TFTP_OBJECTS
vendorProfiles	VENDOR_PROFILES
viewZones	VIEWS_ZONES
TSIGKeys	TSIG_KEYS
GSS	GSS
DHCPZones	DHCP_ZONES
ServerGroup	SERVERGROUP

ENUM Services

Constants used in the ENUM Services.

Property Key	Property Value
H323	H323
SIP	SIP
ifax_mailto	ifax mailto
pres	pres
web_http	web http
web_https	web https
ft_ftp	ft ftp
email_mailto	email mailto
fax_tel	fax tel
sms_tel	sms tel
sms_mailto	sms mailto
ems_tel	ems tel
ems_mailto	ems mailto
mms_tel	mms tel
mms_mailto	mms mailto

Property Key	Property Value
VPIM_MAILTO	VPIM MAILTO
VPIM_LDAP	VPIM LDAP
voice_tel	voice tel
pstn_tel	pstn tel
pstn_sip	pstn sip
xmpp	xmpp
im	im

IP Assignment Action Values

Constants used in the IP Assignment Action Values.

Property Key	Property Value
MAKE_STATIC	MAKE_STATIC
MAKE_RESERVED	MAKE_RESERVED
MAKE_DHCP_RESERVED	MAKE_DHCP_RESERVED

IP Discovery Type

Constants used in the IP address discovery type.

Property Key	Property Value
SNMP	SNMP
PINGSWEEP	PINGSWEEP
SNMP_PINGSWEEP	SNMP_PINGSWEEP
NO_DISCOVERY	NO_DISCOVERY

Object Properties

Constants used in the Object Properties.

Property Key	Property Value
name	name
sharedNetwork	sharedNetwork
CIDR	CIDR
start	start
end	end
template	template
deployable	deployable

Property Key	Property Value
authenticator	authenticator
securityPrivilege	securityPrivilege
historyPrivilege	historyPrivilege
email	email
phoneNumber	phoneNumber
users	users
version	version
description	description
addresses	addresses
address	address
state	state
server	server
serverGroup	serverGroup
serverInterface	serverInterface
zoneTransServerInterface	zoneTransServerInterface
macPool	macPool
view	view
refresh	refresh
retry	retry
expire	expire
minimum	minimum
absoluteName	absoluteName
userAccessType	userAccessType
allowDuplicateHost	allowDuplicateHost
pingBeforeAssign	pingBeforeAssign
defaultDomains	defaultDomains
dnsRestrictions	dnsRestrictions
defaultView	defaultView
comments	comments
ttl	ttl
reverseRecord	reverseRecord
txt	txt
parentZoneName	parentZoneName
linkedParentZoneName	linkedParentZoneName

Property Key	Property Value
linkedRecordName	linkedRecordName
priority	priority
port	port
weight	weight
order	order
preference	preference
service	service
regexp	regexp
replacement	replacement
flags	flags
os	os
cpu	cpu
type	type
rdata	rdata
prefix	prefix
identifier	identifier
parentId	parentId
parentType	parentType
addressIds	addressIds
linkToExternalHost	linkToExternalHost
defaultInterfaceAddress	defaultInterfaceAddress
publishedInterfaceAddress	publishedInterfaceAddress
publishedInterfaceIPv6Address	publishedInterfaceIPv6Address
secondaryServerInterfaceId	secondaryServerInterfaceId
fullHostName	fullHostName
profile	profile
connected	connected
upgrade	upgrade
readOnly	readOnly
servicesIPv4Address	servicesIPv4Address
servicesIPv4Netmask	servicesIPv4Netmask
servicesIPv6Address	servicesIPv6Address
servicesIPv6Subnet	servicesIPv6Subnet
xhaIPv4Address	xhaIPv4Address

Property Key	Property Value
xhIPv4Netmask	xhIPv4Netmask
redundancyScenario	redundancyScenario
xHAServerId	xHAServerId
activeServerId	activeServerId
passiveServerId	passiveServerId
activeServerNewIPv4Address	activeServerNewIPv4Address
activeServerPassword	activeServerPassword
passiveServerPassword	passiveServerPassword
pingAddress	pingAddress
ip6Address	ip6Address
newManagementAddress	newManagementAddress
activeServerIPv4AddressForNAT	activeServerIPv4AddressForNAT
passiveServerIPv4AddressForNAT	passiveServerIPv4AddressForNAT
activeServerNewIPv4AddressForNAT	activeServerNewIPv4AddressForNAT
backboneActiveServerIPv4Address	backboneActiveServerIPv4Address
backboneActiveServerIPv4Netmask	backboneActiveServerIPv4Netmask
backbonePassiveServerIPv4Address	backbonePassiveServerIPv4Address
backbonePassiveServerIPv4Netmask	backbonePassiveServerIPv4Netmask
nodeType	nodeType
breakInProteusOnly	breakInProteusOnly
overrideDHCPValidation	overrideDHCPValidation
checkDHCPConfigurationDeployment	checkDHCPConfigurationDeployment
overrideDNSValidation	overrideDNSValidation
checkDNSConfigurationDeployment	checkDNSConfigurationDeployment
checkDNSZonesDeployment	checkDNSZonesDeployment
postLoadZoneIntegrityValidationDNSDeploy	postLoadZoneIntegrityValidationDNSDeploy
checkNamesValidationModeDNSDeploy	checkNamesValidationModeDNSDeploy
checkIfMXRecordsAreIPsDNSDeploy	checkIfMXRecordsAreIPsDNSDeploy
checkIfMXRecordsPointToCNAMEsDNSDeploy	checkIfMXRecordsPointToCNAMEsDNSDeploy
checkIfNSRecordsAreIPsDNSDeploy	checkIfNSRecordsAreIPsDNSDeploy
checkIfSRVRecordsPointToCNAMEsDNSDeploy	checkIfSRVRecordsPointToCNAMEsDNSDeploy
checkForNonTerminalWildcardsDNSDeploy	checkForNonTerminalWildcardsDNSDeploy
ProteusDDW	ProteusDDW
enableDHCP	enableDHCP

Property Key	Property Value
enableDNS	enableDNS
services	services
importViewName	importViewName
authenticationCredentialDomain	authenticationCredentialDomain
authenticationCredentialUsername	authenticationCredentialUsername
authenticationCredentialPassword	authenticationCredentialPassword
forceDNSFullDeployment	forceDNSFullDeployment
gateway	gateway
reservedAddresses	reservedAddresses
reservedBlock	RESERVED_BLOCK
reservedDHCPRange	RESERVED_DHCP_RANGE
ipGroup	IP_GROUP
templateType	templateType
zoneTemplateType	zonetemplate
IP4NetworkTemplateType	ip4networktemplate
zoneTemplateReapplyMode	zoneReapplyMode
templateReapplyModelgnore	IGNORE
templateReapplyModeUpdate	UPDATE_IF_POSSIBLE
templateReapplyModeOverwrite	OVERWRITE
gatewayReapplyMode	gatewayReapplyMode
reservedAddressesReapplyMode	reservedAddressesReapplyMode
dhcpRangesReapplyMode	dhcpRangesReapplyMode
ipGroupsReapplyMode	ipGroupsReapplyMode
optionsReapplyMode	optionsReapplyMode
noGateway	noGateway
seedRouterAddress	seedRouterAddress
snmpVersion	snmpVersion
snmpPortNumber	snmpPortNumber
snmpCommunityString	snmpCommunityString
securityLevel	securityLevel
context	context
authenticationType	authenticationType
authPassphrase	authPassphrase
privacyPassphrase	privacyPassphrase

Property Key	Property Value
networkBoundaries	networkBoundaries
schedule	schedule
activeStatus	activeStatus
enableLayer2Discovery	enableLayer2Discovery
acceptanceCriteriaReclaim	acceptanceCriteriaReclaim
acceptanceCriteriaUnknown	acceptanceCriteriaUnknown
acceptanceCriteriaMismatch	acceptanceCriteriaMismatch
overrideList	overrideList
matchCriteria	matchCriteria
matchOffset	matchOffset
matchLength	matchLength
customMatchRawString	customMatchRawString
ignoreError	ignoreError
matchValue	matchValue
splitStaticAddresses	splitStaticAddresses
noServerUpdate	noServerUpdate
transientParent	transientParent
optionId	optionId
optionType	optionType
optionAllowMultiple	optionAllowMultiple
optionDescription	optionDescription
deviceTypeId	deviceTypeId
deviceSubTypeId	deviceSubTypeId
ip4Addresses	ip4Addresses
ip6Addresses	ip6Addresses
overrideNamingPolicy	overrideNamingPolicy
deleteKeys	deleteKeys
excludeDHCPRange	excludeDHCPRange
skip	skip
offset	offset
displayName	displayName
hint	hint
accessRight	accessRight
overrideType	overrideType

Property Key	Property Value
retrieveFields	retrieveFields
ignoreCase	ignoreCase
size	size
positionRangeBy	positionRangeBy
positionValue	positionValue
ipGroupBySize	ipGroupBySize
configName	configName
deviceName	deviceName
ipAddressMode	ipAddressMode
ipEntity	ipEntity
viewName	viewName
zoneName	zoneName
recordName	recordName
macAddressMode	macAddressMode
macEntity	macEntity
VCO_MODE_REQUEST_VALUE	REQUEST_VALUE
VCO_MODE_REQUEST_STATIC	REQUEST_STATIC
VCO_MODE_REQUEST_DHCP_RESERVED	REQUEST_DHCP_RESERVED
VCO_MODE_PASS_VALUE	PASS_VALUE
allowDuplicateHosts	allowDuplicateHosts
netmask	netmask
ip	ip
inherited	inherited
redirectTarget	redirectTarget
responsePolicyType	responsePolicyType
workflowLevel	workflowLevel
deploymentAllowed	deploymentAllowed
quickDeploymentAllowed	quickDeploymentAllowed
TraversalMethodology.NO_TRAVERSAL	NO_TRAVERSAL
TraversalMethodology.DEPTH_FIRST	DEPTH_FIRST
TraversalMethodology.BREADTH_FIRST	BREADTH_FIRST
reuseExisting	reuseExisting
secondStandbyServer	secondStandbyServer
nsRecordTTL	nsRecordTTL

Property Key	Property Value
serviceType	serviceType
routerPortInfo	routerPortInfo
switchPortInfo	switchPortInfo
vlanInfo	vlanInfo
discoveryType	discoveryType
pingSweepRange	pingSweepRange
seedRouterAddress	seedRouterAddress
snmpVersion	snmpVersion
snmpPortNumber	snmpPortNumber
snmpCommunityString	snmpCommunityString
secutityLevel	secutityLevel
context	context
authenticationType	authenticationType
authPassphrase	authPassphrase
privacyPassphrase	privacyPassphrase
privacyType	privacyType
networkBoundaries	networkBoundaries
blackHoleVlan	blackHoleVlan
trunkDefaultVlan	trunkDefaultVlan
skipFqdn	skipFqdn
dnsServers	dnsServers
schedule	schedule
ipV4ReconciliationStatus	ipV4ReconciliationStatus
acceptanceCriteriaReclaim	acceptanceCriteriaReclaim
acceptanceCriteriaUnknown	acceptanceCriteriaUnknown
acceptanceCriteriaMismatch	acceptanceCriteriaMismatch
overriddenList	overriddenList
convertOrphanedIPAddressesTo	convertOrphanedIPAddressesTo
locationCode	locationCode
locationInherited	locationInherited
code	code
country	country
subdivision	subdivision
localizedName	localizedName

Property Key	Property Value
description	description
latitude	latitude
longitude	longitude
reserved	reserved
mname	mname
serialNumberFormat	serialNumberFormat
parameterRequestList	parameterRequestList
vendorClassIdentifier	vendorClassIdentifier
macVendor	macVendor
disableDnsOptionInheritance	disableDnsOptionInheritance
leaseTime	leaseTime
expiryTime	expiryTime
deleteOrphanedIPAddresses	deleteOrphanedIPAddresses
ptrs	ptrs

Object Types

Constants used in the Object Types.

Property Key	Property Value
Entity	Entity
Configuration	Configuration
View	View
Zone	Zone
InternalRootZone	InternalRootZone
ZoneTemplate	ZoneTemplate
EnumZone	EnumZone
EnumNumber	EnumNumber
HostRecord	HostRecord
AliasRecord	AliasRecord
MXRecord	MXRecord
TXTRecord	TXTRecord
SRVRecord	SRVRecord
GenericRecord	GenericRecord
HINFORecord	HINFORecord
NAPTRRecord	NAPTRRecord

Property Key	Property Value
RecordWithLink	RecordWithLink
ExternalHostRecord	ExternalHostRecord
StartOfAuthority	StartOfAuthority
IP4Block	IP4Block
IP4Network	IP4Network
IP6Block	IP6Block
IP6Network	IP6Network
IP6Address	IP6Address
IP4NetworkTemplate	IP4NetworkTemplate
DHCP4Range	DHCP4Range
DHCP6Range	DHCP6Range
IP4Address	IP4Address
MACPool	MACPool
DenyMACPool	DenyMACPool
MACAddress	MACAddress
TagGroup	TagGroup
Tag	Tag
User	User
UserGroup	UserGroup
Server	Server
ServerGroup	ServerGroup
NetworkServerInterface	NetworkServerInterface
PublishedServerInterface	PublishedServerInterface
NetworkInterface	NetworkInterface
VirtualInterface	VirtualInterface
LDAP	LDAP
Kerberos	Kerberos
KerberosRealm	KerberosRealm
Radius	Radius
TFTPGroup	TFTPGroup
TFTPFolder	TFTPFolder
TFTPFile	TFTPFile
TFTPDploymentRole	TFTPDploymentRole
DNSDeploymentRole	DNSDeploymentRole

Property Key	Property Value
DHCPDeploymentRole	DHCPDeploymentRole
DNSOption	DNSOption
DHCPV4ClientOption	DHCPV4ClientOption
DHCPServiceOption	DHCPServiceOption
DHCPRawOption	DHCPRawOption
DNSRawOption	DNSRawOption
DHCPV6ClientOption	DHCPV6ClientOption
DHCPV6ServiceOption	DHCPV6ServiceOption
DHCPV6RawOption	DHCPV6RawOption
VendorProfile	VendorProfile
VendorOptionDef	VendorOptionDef
VendorClientOption	VendorClientOption
CustomOptionDef	CustomOptionDef
DHCPMatchClass	DHCPMatchClass
DHCPSubClass	DHCPSubClass
Device	Device
DeviceType	DeviceType
DeviceSubtype	DeviceSubtype
DeploymentScheduler	DeploymentScheduler
IP4ReconciliationPolicy	IP4ReconciliationPolicy
DNSSECSigningPolicy	DNSSECSigningPolicy
IP4IPGroup	IP4IPGroup
ResponsePolicy	ResponsePolicy
TSIGKey	TSIGKey
RPZone	RPZone
Location	Location
InterfaceID	InterfaceID

Option Types

Constants used in the Option Types.

Property Key	Property Value
DNS_RAW	DNS_RAW
DHCP_RAW	DHCP_RAW
DHCPV6_RAW	DHCPV6_RAW

Property Key	Property Value
DNS	DNS
DHCPClient	DHCPClient
DHCPVendorClient	DHCPVendorClient
DHCPService	DHCPService
DHCP6Client	DHCP6Client
DHCP6VendorClient	DHCP6VendorClient
DHCP6Service	DHCP6Service
START_OF_AUTHORITY	START_OF_AUTHORITY

PositionRangeBy

Constants used in the PositionRangeBy API.

Property Key	Property Value
START_OFFSET	START_OFFSET
END_OFFSET	END_OFFSET
START_ADDRESS	START_ADDRESS

Response Policy Type

Constants used in the Response Policy Type.

Property Key	Property Value
BLACKLIST	BLACKLIST
BLACKHOLE	BLACKHOLE
WHITELIST	WHITELIST
REDIRECT	REDIRECT

Response Policy Search Scopes

Constants used in the Response Policy Search.

Property Key	Property Value
RPItemSearchScope.LOCAL	Local
RPItemSearchScope.FEED	Feed
RPItemSearchScope.ALL	All

Reverse Zone Format Type

Constants used in the reverse zone format type.

Property Key	Property Value
STARTIP_NETMASK_NET	"[start-ip]-[net-mask].[net].in-addr.arpa"
STARTIP_ENDIP_NET	"[start-ip]-[end-ip].[net].in-addr.arpa"
STARTIP_SLASH_NETMASK_NET	"[start-ip]/[net-mask].[net].in-addr.arpa"
STARTIP_SLASH_ENDIP_NET	"[start-ip]/[end-ip].[net].in-addr.arpa"
CUSTOM	"custom"

Server Capability Profiles

Constants used in the Server Capability Profiles.

Property Key	Property Value
ADONIS_800	ADONIS_800
ADONIS_1200	ADONIS_1200
ADONIS_1900	ADONIS_1900
ADONIS_1950	ADONIS_1950
ADONIS_XMB2	ADONIS_XMB2
ADONIS_XMB3	ADONIS_XMB3
DNS_DHCP_SERVER_20	DNS_DHCP_SERVER_20
DNS_DHCP_SERVER_45	DNS_DHCP_SERVER_45
DNS_DHCP_SERVER_60	DNS_DHCP_SERVER_60
DNS_DHCP_SERVER_100	DNS_DHCP_SERVER_100
DNS_DHCP_SERVER_100_D	DNS_DHCP_SERVER_100_D
AFILIAS_DNS_SERVER	AFILIAS_DNS_SERVER
OTHER_DNS_SERVER	OTHER_DNS_SERVER
PROTEUS_DDW	PROTEUS_DDW
WINDOWS_SERVER	WINDOWS_SERVER

Service Types

Constants used in the Service Types.

Property Key	Property Value
DNS	DNS
DHCP	DHCP

SNMP Version

Constants used in the SNMP Version.

Property Key	Property Value
V1	v1
V2C	v2c
V3	v3

SNMP Security Levels

Constants used in the SNMP Security Levels.

Property Key	Property Value
AUTH_PRIV	AUTH_PRIV
AUTH_NOPRIV	AUTH_NOPRIV
NOAUTH_NOPRIV	NOAUTH_NOPRIV

SNMP Authentication Type

Constants used in the SNMP Authentication Type.

Property Key	Property Value
MD5	MD5
SHA	SHA

SNMP Privacy Type

Constants used in the SNMP Privacy Type.

Property Key	Property Value
DES	DES
AES128	AES128
AES192	AES192
AES256	AES256

Traversal Methodology

Constants used in the Traversal Methodology.

Property Key	Property Value
TraversalMethodology.NO_TRAVERSAL	NO_TRAVERSAL
TraversalMethodology.DEPTH_FIRST	DEPTH_FIRST
TraversalMethodology.BREADTH_FIRST	BREADTH_FIRST

User Access Type

Constants used in the User Access Type.

Property Key	Property Value
GUI	GUI
API	API
GUI_AND_API	GUI_AND_API

User-defined Field Type

Constants used in the User-defined Field Type.

Property Key	Property Value
TEXT	TEXT
DATE	DATE
BOOLEAN	BOOLEAN
INTEGER	INTEGER
LONG	LONG
EMAIL	EMAIL
URL	URL

User-defined Field Validator Properties

Constants used in the User-defined Field Validator Properties.

Property Key	Property Value
MIN	min
MAX	max
MIN_LENGTH	minLength
MAX_LENGTH	maxLength
PATTERN	pattern

User History Privileges

Constants used in the User History Privileges.

Property Key	Property Value
HIDE	HIDE
VIEW_HISTORY_LIST	VIEW_HISTORY_LIST

User Security Privileges

Constants used in the User Security Privileges.

Property Key	Property Value
NO_ACCESS	NO_ACCESS
VIEW_MY_ACCESS_RIGHTS	VIEW_MY_ACCESS_RIGHTS
VIEW_OTHERS_ACCESS_RIGHTS	VIEW_OTHERS_ACCESS_RIGHTS
CHANGE_ACCESS_RIGHTS	CHANGE_ACCESS_RIGHTS
ADD_ACCESS_RIGHTS	ADD_ACCESS_RIGHTS
DELETE_ACCESS_RIGHTS	DELETE_ACCESS_RIGHTS

User Type

Constants used in the User Type.

Property Key	Property Value
ADMIN	ADMIN
REGULAR	REGULAR

Vendor Profile Option Types

Constants used in the Vendor Profile Option Types.

Property Key	Property Value
IP4	IP4
TEXT	TEXT
UNSIGNED_INT_8	UNSIGNED_INT_8
UNSIGNED_INT_16	UNSIGNED_INT_16
UNSIGNED_INT_32	UNSIGNED_INT_32
UNSIGNED_INT_64	UNSIGNED_INT_64
SIGNED_INT_8	SIGNED_INT_8
SIGNED_INT_16	SIGNED_INT_16
SIGNED_INT_32	SIGNED_INT_32
BOOLEAN	BOOLEAN
IP4_MASK	IP4_MASK
STRING	STRING
BINARY	BINARY
ENCAPSULATED	ENCAPSULATED

Workflow Levels

Constants used in the Workflow Levels.

Property Key	Property Value
None	NONE
Recommend	RECOMMEND
Approve	APPROVE

Defined Probe Values

Constants used in the Data collection process.

Property Key	Property Value
LEASE_COUNT_PER_DATE	LEASE_COUNT_PER_DATE
NETWORK_BLOOM	NETWORK_BLOOM

Probe Status Values

Constants used to check the status of the Data collection process.

Property Key	Property Value
INIT	0
INQUEUE	1
PROCESSING	2
COMPLETED	3

API Method Reference

Topics:

- [*API Sessions*](#)
- [*Generic Methods*](#)
- [*User-defined Fields*](#)
- [*IPAM*](#)
- [*DHCP*](#)
- [*DNS*](#)
- [*Deployment Options*](#)
- [*TFTP*](#)
- [*Servers and Deployment*](#)
- [*Crossover High Availability \(XHA\)*](#)
- [*Address Manager Objects*](#)

This chapter provides the reference table for all methods available in the Address Manager API.

API Sessions

<i>Log in and Log out</i>	login (String name, String password), logout()
<i>System Information</i>	String getSystemInfo()

Generic Methods

<i>Updating Objects</i>	void update (APIEntity <i>entity</i>) All extensions of this method in this table list only mutable parameters.
<i>Update with Options</i>	void updateWithOptions (APIEntity <i>entity</i> , String <i>options</i>)
<i>Deleting Objects</i>	void delete (long <i>objectId</i>)
<i>Delete with Options</i>	void deleteWithOptions (long <i>objectId</i> , String <i>options</i>)
<i>Get Entity by Name</i>	APIEntity getEntityByName (long <i>parentId</i> , String <i>name</i> , String <i>type</i>)
<i>Get Entity by ID</i>	APIEntity getEntityById (long <i>id</i>)
<i>Get Entities</i>	APIEntity[] getEntities (long <i>parentId</i> , String <i>type</i> , int <i>start</i> , int <i>count</i>)
<i>Get Parent</i>	APIEntity[] getParent(long <i>entityId</i>)
<i>Get Entities by Name</i>	APIEntity[] getEntitiesByName (long <i>parentId</i> , String <i>name</i> , String <i>type</i> , int <i>start</i> , int <i>count</i>)
<i>Get Entities by Name Using Options</i>	APIEntity[] getEntitiesByNameUsingOptions (long <i>parentId</i> , String <i>name</i> , String <i>type</i> , int <i>start</i> , int <i>count</i> , String <i>options</i>)
<i>Get MAC Address</i>	APIEntity[] getMACAddress (long <i>configurationId</i> , String <i>macAddress</i>)
<i>Get Linked Entities</i>	APIEntity[] getLinkedEntities (long <i>entityId</i> , String <i>type</i> , int <i>start</i> , int <i>count</i>)
<i>Custom Search</i>	APIEntity[] customSearch (String[] <i>filters</i> , String <i>type</i> , String[] <i>options</i> , Integer <i>start</i> , Integer <i>count</i>)
<i>Search by Category</i>	APIEntity[] searchByCategory (String <i>keyword</i> , String <i>category</i> , int <i>start</i> , int <i>count</i>)
<i>Search by Object Types</i>	APIEntity[] searchByObjectTypes (String <i>keyword</i> , String <i>types</i> , int <i>start</i> , int <i>count</i>)

Linked Entities

<i>Link Entities</i>	void linkEntities (long <i>entity1Id</i> , long <i>entity2Id</i> , String <i>properties</i>)
----------------------	--

<i>Unlink Entities</i>	void unlinkEntities (long entity1Id, long entity2Id, String properties)
------------------------	---

Changing Locale

<i>Log in with Options</i>	void loginWithOptions(String userName, String password, String options)
----------------------------	---

User-defined Fields

<i>Get User-defined Field</i>	APIUserDefinedField[] getUserDefinedFields (String type, boolean requiredFieldsOnly)
<i>Update Bulk User-defined Field</i>	byte[] updateBulkUdf (byte[] data, String properties)

IPAM

IPv4 Blocks

<i>Add IPv4 Block by CIDR</i>	long addIPv4BlockByCIDR (long parentId, String CIDR, String properties)
<i>Add IPv4 Block by Range</i>	long addIPv4BlockByRange (long parentId, String start, String end, String properties)
<i>Add Parent Block</i>	void addParentBlock (long[] blockOrNetworkIDs)
<i>Add Parent Block with Properties</i>	long addParentBlockWithProperties (long[] blockOrNetworkIDs String properties)
<i>Get IP Range by IP Address</i>	APIEntity getIPRangedByIP (long containerId, String type, String address)
<i>Get IPv4 Block by CIDR</i>	APIEntity getEntityByCIDR (long parentId, String cidr, String type)
<i>Get IPv4 Block by Range</i>	APIEntity getEntityByRange (long parentId, String address1, String address2, String type)
<i>Merge Blocks with Parent</i>	void mergeBlocksWithParent (long[] blockIDs)
<i>Merge Selected Blocks or Networks</i>	void mergeSelectedBlocksOrNetworks (long[] blockOrNetworkIds, long blockOrNetworkToKeep)
<i>Move IPv4 Object</i>	void moveIPv4Object (long objectId, String address)
<i>Move IP Object</i>	void moveIPv4Object (long objectId, String address, String options)
<i>Resize Range</i>	void resizeRange (long objectId, String range, String options)

<i>Update IPv4 Block</i>	void update (APIEntity entity) For more information, see generic update() method.
<i>IPv4 Block Generic Methods</i>	void delete (long objectId)

IPv4 Networks

<i>Add IPv4 Network</i>	long addIP4Network (long blockId, String CIDR, String properties)
<i>Get IPv4 Range by IP Address</i>	APIEntity getIPRangedByIP (long containerId, String type, String address)
<i>Get IPv4 Network by CIDR</i>	APIEntity getEntityByCIDR (long parentId, String cidr, String type)
<i>Get IPv4 Network by Hint</i>	APIEntity[] getIP4NetworksByHint (long containerId, int start, int count, String options)
<i>Get IPv4 Network by Range</i>	APIEntity getEntityByRange (long parentId, String address1, String address2, String type)
<i>Get Next Available Network</i>	long getNextAvailableIP4Network (long parentId, long size, boolean isLargerAllowed, boolean autoCreate)
<i>Get Next Available IP Range</i>	APIEntity getNextAvailableIPRange (long parentId, long size, String type, String properties)
<i>Get Next Available IP Ranges</i>	APIEntity[] getNextAvailableIPRanges (long parentId, long size, String type, int count, String properties)
<i>Split IPv4 Network</i>	APIEntity[] splitIP4Network (long networkId, int numberOfParts, String options)
<i>Merge Selected Blocks or Networks</i>	void mergeSelectedBlocksOrNetworks (long[] blockOrNetworkIds, long blockOrNetworkToKeep)
<i>Move IPv4 Object</i>	void moveIP4Object (long objectId, String address)
<i>Move IP Object</i>	void moveIP4Object (long objectId, String address, String options)
<i>Resize Range</i>	void resizeRange (long objectId, String range String options)
<i>Update IPv4 Network</i>	void update (APIEntity entity) For more information, see generic update() method.
<i>IPv4 Network Generic Methods</i>	void delete (long objectId)

IPv4 Network Templates

<i>Update IPv4 Network Template Name</i>	void update (APIEntity entity) For more information, see generic update() method.
<i>IPv4 Network Template Generic Methods</i>	get() void delete(long objectId)
<i>Add IPv4 Network Template</i>	long addIP4NetworkTemplate (long configurationId, String name, String properties)
<i>Assign or Update Template</i>	void assignOrUpdateTemplate (long entityId, long templateId, String properties)
<i>Re-apply Template</i>	void reapplyTemplate (long templateId, String properties)

IPv4 Addresses

<i>Assign IPv4 Address</i>	long assignIP4Address (long configurationId, String ip4Address, String macAddress, String hostInfo, String action, String properties)
<i>Assign Next Available IPv4 Address</i>	APIEntity assignNextAvailableIP4Address (long configurationId, long parentId, String macAddress, String hostInfo, String action, String properties)
<i>Get IPv4 Address</i>	APIEntity getIP4Address (long containerId, String address)
<i>Get Next IPv4 Address</i>	String getNextIP4Address (long parentId, String properties)
<i>Check Allocation for IPv4 Address</i>	boolean isAddressAllocated (long configurationId, String ipAddress, String macAddress)
<i>Get Next Available Address</i>	String getNextAvailableIP4Address (long parentId)
<i>Update IPv4 Address</i>	void update (APIEntity entity) For more information, see generic update() method.
<i>IPv4 Address Generic Methods</i>	void delete (long objectId)
<i>Change IPv4 Address State</i>	void changeStateIP4Address (long addressId, String targetState, String macAddress)

IPv4 Objects

<i>Move IPv4 Object</i>	void moveIP4Object(long objectId, String address)
-------------------------	---

<i>Move IP Object</i>	void moveIP4Object(long objectId, String address, String options)
<i>Resize Range</i>	void resizeRange(long objectId, String range, String options)

IPv4 Group

<i>Add IPv4 IP Group by Range</i>	long addIP4IPGroupByRange(long parentId, String start, String end, String properties)
<i>Add IPv4 IP Group by Size</i>	long addIP4IPGroupBySize(long parentId, String name, int size, String positionRangeBy, String positionValue, String properties)

IPv4 Discovery and Reconciliation

<i>Add IPv4 Reconciliation Policy</i>	long addIP4ReconciliationPolicy (long parentId, String name, String properties)
<i>Get Discovered Devices</i>	APIEntity[] getDiscoveredDevices (long policyId)
<i>Get Discovered Device</i>	APIEntity getDiscoveredDevice (long policyId, long deviceId)
<i>Get Discovered Device Interfaces</i>	APIEntity getDiscoveredDeviceInterfaces (long policyId, long deviceId)
<i>Get Discovered Device Networks</i>	APIEntity getDiscoveredDeviceNetworks (long policyId, long deviceId)
<i>Get Discovered Device Hosts</i>	APIEntity getDiscoveredDeviceHosts (long policyId, long deviceId)
<i>Get Discovered Device Vlans</i>	APIEntity getDiscoveredDeviceVlans (long policyId, long deviceId)
<i>Get Discovered Device ARP Entries</i>	APIEntity getDiscoveredDeviceArpEntries (long policyId, long deviceId)
<i>Get Discovered Device MAC Address Entries</i>	APIEntity getDiscoveredDeviceMacAddressEntries (long policyId, long deviceId)

IPv6 Objects

<i>Add IPv6 Address</i>	long addIP6Address(long containerId, String address, String type, String name, String properties)
<i>Add IPv6 Block by MAC Address</i>	long addIP6BlockByMACAddress(long parentId, String macAddress, String name, String properties)
<i>Add IPv6 Block by Prefix</i>	long addIP6BlockByPrefix(long parentId, String prefix, String name, String properties)
<i>Add IPv6 Network by Prefix</i>	long addIP6NetworkByPrefix(long parentId, String prefix, String name, String properties)

<i>Split IPv6 Block or Network</i>	APIEntity[] splitIP6Range (long rangeId, int numberOfParts, String options)
<i>Get IPv6 Range by IP Address</i>	APIEntity getIPRangedByIP(long containerId, String type, String address)
<i>Get IPv6 Network by Hint</i>	APIEntity getIP6ObjectsByHint (long containerId, String objectType, int start, int count, String options)
<i>Assign IPv6 Address</i>	boolean assignIP6Address(long containerId, String address, String action, String macAddress, String hostInfo, String properties)
<i>Clear IPv6 Address</i>	boolean clearIP6Address(long addressId)
<i>Get Entity by Prefix</i>	APPIEntity getEntityByPrefix(long containerId, String prefix, String type)
<i>Get IPv6 Address</i>	APPIEntity getIP6Address(long containerId, String address)
<i>Reassign IPv6 Address</i>	long reassignIP6Address(long oldAddressId, String destination, String properties)

Provision Devices

<i>Add Device Instance</i>	String addDeviceInstance (String configName, String deviceName, String ipAddressMode, String ipEntity, String viewName, String zoneName, String recordName, String macAddressMode, String macEntity, String options)
<i>Delete Device Instance</i>	void deleteDeviceInstance (String configName, String identifier, String options)

DHCP

IPv4 DHCP Ranges

<i>Add IPv4 DHCP Range</i>	long addDHCP4Range(long networkId, String start, String end, String properties)
<i>Add IPv4 DHCP Range By Size</i>	long addDHCP4RangeBySize(long networkId, String offset, String size, String properties)
<i>Get IPv4 Range by IP Address</i>	APIEntity getIPRangedByIP(long containerId, String type, String address)
<i>Get IPv4 DHCP Range</i>	APIEntity getEntityByRange(long parentId, String address1, String address2, String type)
<i>Get IPv4 DHCP Ranges</i>	APIEntity[] getEntities(long parentId, String type, int start, int count)
<i>Get Max Allowed Range</i>	String[] getMaxAllowedRange(long rangeId)

<i>Update IPv4 DHCP Range</i>	void update(APIEntity entity)For more information, see generic update() method.
<i>IPv4 DHCP Range Generic Methods</i>	void delete(long objectId)

IPv6 DHCP Ranges

<i>Add IPv6 DHCP Range</i>	long addDHCP6Range(long networkId, String start, String end, String properties)
<i>Add IPv6 DHCP Range by Size</i>	long addDHCP6RangeBySize(long networkId, String start, String size, String properties)
<i>Get IPv6 Range by IP Address</i>	APIEntity getIPRangedByIP(long containerId, String type, String address)
<i>Get IPv6 DHCP Range</i>	APIEntity getEntityByRange(long parentId, String address1, String address2, String type)
<i>Get Multiple IPv6 DHCP Ranges</i>	APIEntity[] getEntities(long parentId, String type, int start, int count)
<i>Update IPv6 DHCP Range</i>	void update(APIEntity entity)For more information, see generic update() method.
<i>IPv6 DHCP Range Generic Methods</i>	void delete(long objectId)

DHCP Client Options

<i>Add DHCP Client Option</i>	long addDHCPClientDeploymentOption(long entityId, String name, String value, String properties)
<i>Get DHCP Client Option</i>	APIDeploymentOption getDHCPClientDeploymentOption(long entityId, String name, long serverId)
<i>Update DHCP Client Option</i>	void updateDHCPClientDeploymentOption (APIDeploymentOption option)
<i>Delete DHCP Client Option</i>	void deleteDHCPClientDeploymentOption(long entityId, String name, long serverId)

DHCP6 Client Options

<i>Add DHCP6 Client Option</i>	long addDHCP6ClientDeploymentOption(long entityId, String name, String value, String properties)
<i>Get DHCP6 Client Option</i>	APIDeploymentOption getDHCP6ClientDeploymentOption(long entityId, String name, long serverId)
<i>Update DHCP6 Client Option</i>	void updateDHCP6ClientDeploymentOption (APIDeploymentOption option)

<i>Delete DHCP6 Client Option</i>	<code>void deleteDHCP6ClientDeploymentOption(long entityId, String name, long serverId)</code>
-----------------------------------	--

DHCP Custom Options

<i>Add Custom Deployment Option</i>	<code>long addCustomOptionDefinition(long configurationId, String name, long optionId, String optionType, boolean allowMultiple, String properties)</code>
-------------------------------------	--

DHCP Service Options

<i>Add DHCP Service Option</i>	<code>long addDHCPServiceDeploymentOption(long entityId, String name, String value, String properties)</code>
<i>Get DHCP Service Option</i>	<code>APIDeploymentOption getDHCPServiceDeploymentOption(long entityId, String name, long serverId)</code>
<i>Update DHCP Service Option</i>	<code>void updateDHCPServiceDeploymentOption(APIDeploymentOption option)</code>
<i>Delete DHCP Service Option</i>	<code>void deleteDHCPServiceDeploymentOption(long entityId, String name, long serverId)</code>

DHCP6 Service Options

<i>Add DHCP6 Service Option</i>	<code>long addDHCP6ServiceDeploymentOption(long entityId, String name, String value, String properties)</code>
<i>Get DHCP6 Service Option</i>	<code>APIDeploymentOption getDHCP6ServiceDeploymentOption(long entityId, String name, long serverId)</code>
<i>Update DHCP6 Service Option</i>	<code>void updateDHCP6ServiceDeploymentOption(APIDeploymentOption option)</code>
<i>Delete DHCP6 Service Option</i>	<code>void deleteDHCP6ServiceDeploymentOption(long entityId, String name, long serverId)</code>

DHCP Vendor Options

<i>Add DHCP Vendor Deployment Option</i>	<code>long addDHCPVendorDeploymentOption(long parentId, long optionId, String value, String properties)</code>
<i>Add Vendor Option Definition</i>	<code>long addVendorOptionDefinition(long vendorProfileId, long optionId, String name, String optionType, String description, boolean allowMultiple, String properties)</code>

<i>Add Vendor Profile</i>	long addVendorProfile(String identifier, String name, String description, String properties)
<i>Delete DHCP Vendor Deployment Option</i>	void deleteDHCPVendorDeploymentOption(long entityId, long optionId, long serverId)
<i>Get DHCP Vendor Deployment Option</i>	APIDeploymentOption getDHCPVendorDeploymentOption(long entityId, long optionId, long serverId)
<i>Update DHCP Vendor Deployment Option</i>	void updateDHCPVendorDeploymentOption (APIDeploymentOption option)

DHCP Match Classes

<i>Add DHCP Match Classes</i>	long addDHCPMatchClass (long configurationId, String name, String matchCriteria, String properties)
<i>Update DHCP Match Classes</i>	void update (APIEntity entity) For more information, see generic update() method.
<i>Delete DHCP Match Classes</i>	void delete (long objectId)
<i>Add DHCP Sub Classes</i>	long addDHCPSubClass (long matchClassId, String matchValue, String properties)
<i>Update DHCP Sub Classes</i>	void update (APIEntity entity) For more information, see generic update() method.
<i>Delete DHCP Sub Classes</i>	void delete (long objectId)

Shared Networks

<i>Link a Shared Network Tag</i>	void shareNetwork (long networkId, long tagId)
<i>Unlink a Shared Network Tag</i>	void unshareNetwork (long networkId)
<i>Get Shared Networks</i>	APIEntity[] getSharedNetworks (long tagId)

DNS

DNS Views

<i>Add DNS View</i>	long addView(long configurationId, String name, String properties)
<i>Update DNS View</i>	void update(APIEntity entity)For more information, see generic update() method.
<i>DNS View Generic Methods</i>	getEntity()void delete(long objectId)

Add Access Control List (ACL)	long addACL(long configurationId, String name, String properties)
Update Access Control List (ACL)	void update(APIEntity entity)For more information, see Updating Objects on page 52.

DNS Zones

Add Entity for DNS Zones	long addEntity(long parentId, APIEntity entity)
Add Zone	long addZone(long parentId, String absoluteName, String properties)
Get Zones by Hint	APIEntity[] getZonesByHint(long containerId, int start, int count, String options)
Update Zone	void update(APIEntity entity)For more information, see generic update() method.
Zone Generic Methods	getEntity()void delete(long objectId)
Get Key Signing Key	String[] getKSK (long entityId, String format)

DNS Zone Templates

Add Zone Template	long addZoneTemplate(long parentId, String name, String properties)
Assign or Update Template	void assignOrUpdateTemplate(long entityId, long templateId, String properties)
Update Zone Template	void update(APIEntity entity) For more information, see generic update() method.
Zone Template Generic Methods	getEntity()void delete(long objectId)
Add Records to DNS Zone Template	addEntity()

ENUM Zones

Add ENUM Zone	long addEnumZone(long parentId, long prefix, String properties)
Update ENUM Zone	void update(APIEntity entity)For more information, see generic update() method.
ENUM Zone Generic Methods	getEntity()void delete(long objectId)

ENUM Numbers

Add ENUM Number	long addEnumNumber(long parentId, int number, String properties)
Update ENUM Number	void update(APIEntity entity)For more information, see generic update() method.

<i>ENUM Number Generic Methods</i>	getEntity()void delete(long objectId)
------------------------------------	---

Generic Resource Records

<i>Add Resource Record</i>	long addResourceRecord(long viewId, String absoluteName, String type, String rdata, long ttl, String properties)
<i>Add Entity for Resource Records</i>	long addEntity(long parentId, APIEntity entity)
<i>Move Resource Records</i>	void moveResourceRecord(long resourceRecordId, String destinationZone)

NAPTR Records

<i>Add NAPTR Record</i>	long addNAPTRRecord(long viewId, String absoluteName, int order, int preference, String service, String regexp, String replacement, String flags, long ttl, String properties)
<i>Update NAPTR Record</i>	update(int order, int preference, String service, String regexp, String replacement, long ttl)For more information, see generic update() method.
<i>NAPTR Record Generic Methods</i>	getEntity()void delete(long objectId)

External Host Records

<i>Add External Host Record</i>	long addExternalHostRecord(long viewId, String name, String properties)
<i>Update External Host Record</i>	void update(APIEntity entity)For more information, see generic update() method.
<i>External Host Record Generic Methods</i>	getEntity()void delete(long objectId)

Host Records

<i>Add Host Record</i>	long addHostRecord(long viewId, String absoluteName, String addresses, long ttl, String properties)
<i>Add Bulk Host Records</i>	APIEntity[] addBulkHostRecord (long viewId, String absoluteName, long ttl, long networkId, String startAddress, int numberOfAddresses, String properties)
<i>Get Host Record by Hint</i>	APIEntity[] getHostRecordsByHint (int start, int count, String options)
<i>Get IP Address with Host Records</i>	APIEntity[] getNetworkLinkedProperties(long networkId)

<i>Get Dependent Records</i>	APIEntity[] getDependentRecords(long entityId, int start, int count)
<i>Update Host Record</i>	void update(String addresses, long ttl, String comment)For more information, see generic update() method.
<i>Host Record Generic Methods</i>	getEntity()void delete(long objectId)

Alias Records

<i>Add Alias Record</i>	long addAliasRecord(long viewId, String absoluteName, String linkedRecordName, long ttl, String properties)
<i>Get Aliases by Hint</i>	APIEntity[] getAliasesByHint (int start, int count, String options)
<i>Update Alias Record</i>	void update(String linkedRecordName, long ttl, String comment)For more information, see generic update() method.
<i>Alias Record Generic Methods</i>	getEntity()void delete(long objectId)

Text Records

<i>Add Text Record</i>	long addTXTRecord(long viewId, String absoluteName, String txt, long ttl, String properties)
<i>Update Text Record</i>	void update(long ttl, String comment String txt)For more information, see generic update() method.
<i>Text Record Generic Methods</i>	getEntity()void delete(long objectId)

HINFO Records

<i>Add HINFO Record</i>	long addHINFORecord(long viewId, String absoluteName, String cpu, String os, long ttl, String properties)
<i>Update HINFO Record</i>	void update(long ttl, String comment String cpu, String os)For more information, see generic update() method.
<i>HINFO Record Generic Methods</i>	getEntity()void delete(long objectId)

MX Records

<i>Add MX Record</i>	long addMXRecord(long viewId, String absoluteName, int priority, String linkedRecordName, long ttl, String properties)
----------------------	--

<i>Update MX Record</i>	void update(String linkedRecordName, long ttl, int priority, String comment)For more information, see generic update() method.
<i>MX Record Generic Methods</i>	getEntity()void delete(long objectId)

SRV Records

<i>Add SRV Record</i>	long addSRVRecord(long viewId, String absoluteName, int priority, int port, int weight, String linkedRecordName, long ttl, String properties)
<i>Update SRV Record</i>	void update(String linkedRecordName, long ttl, int priority, int port, int weight, String comment)For more information, see generic update() method.
<i>SRV Record Generic Methods</i>	getEntity()void delete(long objectId)

Start of Authority Records

<i>Add Start of Authority Record</i>	long addStartOfAuthority(long parentId, String email, long refresh, long retry, long expire, long minimum, String properties)
<i>Update Start of Authority Record</i>	void update(String email, long refresh, long retry, long expire, long minimum)For more information, see generic update() method.
<i>Start of Authority Record Generic Methods</i>	getEntity()void delete(long objectId)

Generic Records

<i>Add Generic Record</i>	long addGenericRecord(long viewId, String absoluteName, String type, String rdata, long ttl, String properties)
<i>Update Generic Record</i>	void update(String type, String rdata, long ttl, String comment)For more information, see generic update() method.
<i>Generic Record Generic Methods</i>	getEntity()void delete(long objectId)

DNS Options

<i>Add DNS Option</i>	long addDNSDeploymentOption(long entityId, String name, String value, String properties)
<i>Get DNS Option</i>	APIDeploymentOption getDNSDeploymentOption(long entityId, String name, long serverId)

<i>Update DNS Option</i>	void updateDNSDeploymentOption(APIDeploymentOption option)
<i>Delete DNS Option</i>	void deleteDNSDeploymentOption(long entityId, String name, long serverId)

DNS Response Policies

<i>Add Response Policy</i>	long addResponsePolicy(long configurationId, String name, String responsePolicyType, long ttl, String properties)
<i>Upload Response Policy Item</i>	void uploadResponsePolicyItems(long parentId, byte[] policyItemsData)
<i>Search Response Policies</i>	ResponsePolicySearchResult[] searchResponsePolicyItems(String keyword, String scope, int start, int count, String properties)

Reverse Zone Name Format

<i>Add Reverse Zone Name Format</i>	long addDNSDeploymentOption(long entityId, String name, String value, String properties)
-------------------------------------	--

Deployment Options

<i>Get Deployment Options</i>	APIDeploymentOption[] getDeploymentOptions(long entityId, String optionTypes, long serverId)
<i>Add Raw Deployment Option</i>	long addRawDeploymentOption (long parentId, String optionType, String rawData, String properties)
<i>Update Raw Deployment Option</i>	void updateRawDeploymentOption (APIDeploymentOption option)

TFTP

TFTP Groups

<i>Add TFTP Group</i>	long addTFTPGroup(long configurationId, String name, String properties)
<i>Update TFTP Group</i>	void update(APIEntity entity)For more information, see generic update() method.
<i>TFTP Group Generic Methods</i>	getEntity()void delete(long objectId)

TFTP Folders

Add TFTP Folder	long addTFTPFolder(long parentId, String name, String properties)
Update TFTP Folder	void update(APIEntity entity)For more information, see generic update() method.
TFTP Folder Generic Methods	getEntity()void delete(long objectId)

TFTP Files

Add TFTP File	long addTFTPFile(long parentId, String name, String version, byte[] data, String properties)
Update TFTP File	void update(String name, String version, byte[] data, String description)For more information, see generic update() method.
TFTP File Generic Methods	getEntity()void delete(long objectId)

Servers and Deployment

Servers

Add Server	long addServer(long configurationId, string name, string defaultInterfaceAddress, string fullHostName, string profile, string properties)
Import Server	void importServer(long serverId, boolean importDns, boolean importDhcp, string properties)
Replace Server	void replaceServer(long serverId, string name, string defaultInterface, string hostName, string password, boolean upgrade, string properties)
Deploy Server	void deployServer(long serverId)
Deploy Server Configuration	void deployServerConfig (long serverID, String properties)
Deploy Server Services	void deployServerServices(long serverId, String services)
Quick Deployment	void quickDeploy(long zoneld, String properties)
Deployment Status	int getServerDeploymentStatus(long serverId, String properties)
Server Generic Methods	getEntity()void delete(long objectId)

Server Group

Add Server Group	long addEntity(long parentId, APIEntity entity)
Update Server Group	void update (APIEntity entity) For more information, refer to generic <code>update()</code> method.
Server Group Generic Methods	getEntity() void delete(long objectId) For more information, refer to Getting Objects on page 44 and Deleting Objects on page 52.
Add Server to Server Group	void linkEntities (long entity1Id, long entity2Id, String properties) For more information, refer to Link Entities on page 53.
Remove Server from Server Group	void unlinkEntities (long entity1Id, long entity2Id, String properties) For more information, refer to Unlink Entities on page 54.

DNS and DHCP Deployment Roles

Get Servers Associated with a Deployment Role	APIEntity getServerForRole (long roleId)
Get Server's Associated Deployment Roles	APIDeploymentRole[] getServerDeploymentRoles (long serverId)
Get Deployment Roles for DNS and IP Address Space Objects	APIDeploymentRole[] getDeploymentRoles (long entityId)
Move Deployment Roles	moveDeploymentRoles (long sourceServerId, long targetServerInterfaceId, boolean moveDnsRoles, boolean moveDhcpRoles, String options)

DHCP Deployment Roles

Add DHCP Deployment Role	long addDHCPDeploymentRole(long entityId, long serverInterfaceId, String type, String properties)
Get DHCP Deployment Role	APIDeploymentRole getDHCPDeploymentRole(long entityId, long serverInterfaceId)
Update DHCP Deployment Role	void updateDHCPDeploymentRole(APIDeploymentRole role)
Delete DHCP Deployment Role	void deleteDHCPDeploymentRole(long entityId, long serverInterfaceId)

DNS Deployment Roles

<i>Add DNS Deployment Role</i>	long addDNSDeploymentRole(long entityId, long serverInterfaceId, String type, String properties)
<i>Get DNS Deployment Role</i>	APIDeploymentRole getDNSDeploymentRole(long entityId, long serverInterfaceId)
<i>Get DNS Deployment Role for View</i>	APIDeploymentRole getDNSDeploymentRoleForView(long entityId, long serverInterfaceId, long viewId)
<i>Update DNS Deployment Role</i>	void updateDNSDeploymentRole(APIDeploymentRole role)
<i>Delete DNS Deployment Role</i>	void deleteDNSDeploymentRole(long entityId, long serverInterfaceId)
<i>Delete DNS Deployment Role for View</i>	void deleteDNSDeploymentRoleForView(long entityId, long serverInterfaceId, long viewId)

TFTP Deployment Roles

<i>Add TFTP Deployment Role</i>	long addTFTPDeploymentRole(long entityId, long serverId, String properties)
<i>Update TFTP Deployment Role</i>	Not supported
<i>TFTP Deployment Role Generic Methods</i>	getEntity()void delete(long objectId)

Crossover High Availability (XHA)

<i>Create XHA</i>	long createXHAPair(long configurationId, long activeServerId, long passiveServerId, String activeServerNewIPv4Address, String properties)
<i>Edit XHA</i>	void editXHAPair(long xHAServerId, String name, String properties)
<i>Failover XHA</i>	void failoverXHA(long xHAServerId)
<i>Break XHA</i>	void breakXHAPair(long xHAServerId, boolean breakInProteusOnly)

Address Manager Objects

Configurations

<i>Add Configuration</i>	long addEntity(long parentId, APIEntity entity)
<i>Update Configuration</i>	void update(String name, String properties)

	For more information, see generic update() method.
<i>Configuration Generic Methods</i>	getEntity()void delete(long objectId)
<i>Get Configuration Setting</i>	String getConfigurationSetting(long configurationId, String settingName)
<i>Update Configuration Setting</i>	void updateConfigurationSetting(long configurationId, String settingName, String properties)

Groups and Users

<i>Add Group</i>	long addUserGroup(String name, String properties)
<i>Update Group</i>	void update(APIEntity entity)For more information, see generic update() method.
<i>Group Generic Methods</i>	getEntity()void delete(long objectId)
<i>Add User</i>	long addUser(String username, String password, String properties)
<i>Update User</i>	void update(properties = "securityPrivilege=<value> historyPrivilege=<value>")For more information, see generic update() method.
<i>Update User Password</i>	void updateUserPassword(long userId, String newPassword, String[] options)
<i>User Generic Methods</i>	getEntity()void delete(long objectId)

Authenticators

<i>Update Authenticator</i>	void update(APIEntity entity)For more information, see generic update() method.
<i>Authenticator Generic Methods</i>	getEntity()void delete(long objectId)

Access Rights

<i>Add Access Right</i>	long addAccessRight(long entityId, long userId, String value, String overrides)
<i>Get Access Right</i>	APIAccessRight getAccessRight(long entityId, long userId)
<i>Get Access Rights for Entity</i>	APIAccessRight[] getAccessRightsForEntity(long entityId, int start, int count)
<i>Get Access Rights for User</i>	APIAccessRight[] getAccessRightsForUser(long userId, int start, int count)
<i>Update Access Rights</i>	void updateAccessRight(long entityId, long userId, String value String overrides)

Delete Access Rights	void deleteAccessRight(long entityId, long userId)
--------------------------------------	--

Devices

Add Device	long addDevice(long configurationId, String name, long deviceTypeId, long deviceSubtypeId, String ip4Addresses, String ip6Addresses, String properties)
Add Device Subtype	long addDeviceSubtype(long parentId, String name, String properties)
Add Device Type	long addDeviceType(String name, String properties)

Object Tag Groups

Add Object Tag Group	long addTagGroup(String name, String properties)
Update Object Tag Group	void update(APIEntity entity)For more information, see generic update() method.
Object Tag Group Generic Methods	getEntity()void delete(long objectId)

Object Tags

Add Object Tag	long addTag (long parentid, String name, String properties)
Assign Object Tag	This method is deprecated. Using this method now returns an error message. Use the linkEntities() method instead. For more information, see Link Entities on page 50.
Remove Object Tag	This method is deprecated. Using this method now returns an error message. Use the unlinkEntities() method instead. For more information, see Unlink Entities on page 51.
Update Object Tag	void update (APIEntity entity) For more information, see generic update() method.
Object Tag Generic Methods	getEntity() void delete (long objectId)

Locations

Add a Location	long addLocation (String name , long parentId , String properties)
Get Location By Code	APIEntity[] getLocations (long parentId)

[Get All Used Locations](#)APIEntity getLocationByCode (String **code**)

Database Management

[Configure Replication](#)

void configureReplication (String standbyServer, boolean compressReplication, long replicationQueueThreshold, long replicationBreakThreshold, String properties)

[Purge History](#)

int purgeHistoryNow(String untilWhenTimestamp, int numberOfDaysToKeep, int numberOfMonthsToKeep, boolean waitOption)

MAC Pools

[Get MAC Addresses in Pool](#)

This method is deprecated. Using this method now returns an error message. Use the getLinkedEntities() method instead. For more information, see Get Linked Entities on page 49.

[Add MAC Pool](#)

long addEntity(long parentId, APIEntity entity)

[Update MAC Pool](#)

void update(APIEntity entity)For more information, see generic update() method.

[MAC Pool Generic Methods](#)

getEntity()void delete(long objectId)

MAC Addresses

[Add MAC Address](#)

long addMACAddress(long configurationId, String macAddress, String properties)

[Associate MAC Address](#)

void associateMACAddressWithPool(long configurationId, String macAddress, long poolId)

[Deny MAC Address](#)

void denyMACAddress(long configurationId, String macAddress)

[Is Address Allocated?](#)

boolean isAddressAllocated(long configurationId, String ipAddress, String macAddress)

[Update MAC Address](#)

void update(String name, String macpoolId)For more information, see generic update() method.

[MAC Address Generic Methods](#)

getEntity()void delete(long objectId)

[Get MAC Address](#)

APIEntity getMACAddress(long configurationId, String macAddress)

Workflow Change Requests

[Workflow Change Requests](#)

For more information, see Access Rights on page 195.

Migration

<i>Migrate a File</i>	void migrateFile(String filename)
<i>Migration Status</i>	boolean isMigrationRunning(String filename)

Collecting Data

<i>Start Probe</i>	void startProbe (String definedProbe, String properties)
<i>Get Probe Status</i>	int getProbeStatus (String definedProbe)
<i>Get Probe Data</i>	APIData getProbeData (String definedProbe, String properties)

Property Options Reference

Topics:

- [Property Options](#)
- [IP Address States](#)

This chapter provides the lists of available properties and IP address states.

Property Options

This chapter provides the reference table for the available properties that can be updatable or read-only when using the *get*, *add* or *update* API methods. The properties marked with read-only cannot be updated when committing *add* or *update* methods. Refer to these tables to find what value of properties will be returned and what values can be updated.


Configuration

Object Type	Properties	Read-only/Updatable
Configuration	None	None

Views and Zones

Object Type	Properties	Read-only/Updatable
View	None	None
Zone	deployable	Both
Zone Template	None	None
EnumZone	deployable	Both
Response Policy	None	None
EnumNumber	name	Both
	data	Both

Resource Records

Object Type	Properties	Read-only/Updatable
Host Record	ttl=time-to-live value	Both
	absoluteName=the FQDN for the host record	Read-only
	addresses=a list of comma-separated IP addresses (For example: 10.0.0.5,130.4.5.2)	Both
	reverseRecord	Both
Alias Record	ttl=time-to-live value	Both
	absoluteName=the FQDN for the host record	Read-only
	linkedRecordName=the name of the record to which this alias will link.	Both
External Host	addresses=a list of comma-separated IP addresses.	Read-only
	 Note: The External Host Record API entity returns the <i>addresses</i> property only when it is linked with an IP address. If there is no linked IP address, it will return <i>null</i> .	

Object Type	Properties	Read-only/ Updatable
Generic Record	ttl = time-to-live value	Both
	absolutName = the FQDN for the host record	Read-only
	type = Resource record type (For example: A/AAAA/PTR/SRV/MX)	Read-only
	rdata = Resource record data (comma-separated values as per the record type)	Both
Host Info Record	ttl = time-to-live value	Both
	absolutName = the FQDN for the host record	Read-only
	os = a string providing operation system information	Both
	cpu = a string providing central processing unit information	Both
Mail Exchanger Record	ttl = time-to-live value	Both
	absolutName = the FQDN for the host record	Read-only
	linkedRecordName = the FQDN of the host record to which this MX record is linked	Both
	priority = specifies which mail server to send clients to first when multiple matching MX records are present. Multiple MX records with equal priority values are referred to in a round-robin fashion.	Both
Naming Authority Pointer	ttl = time-to-live value	Both
	absolutName = the FQDN for the host record	Read-only
	order = specifies the order in which NAPTR records are read, if several records are present and are possible matches. The lower order value takes precedence.	Both
	preference = specifies the order in which NAPTR records are read if the order values are the same in multiple records. The lower preference value takes precedence.	Both
	service = specifies the service used for the NAPTR record	Both
	regexp = a regular expression, enclosed in double quotation marks, used to transform the client data. If a regular expression is not specified, a domain name must be specified in the replacement parameter.	Both
	replacement = specifies a domain name as an alternative to the regexp. This parameter replaces client data with a domain name	Both
	flags = an optional parameter used to set flag values for the record.	Both
Service Record	ttl = time-to-live value	Both
	absolutName = the FQDN for the host record	Read-only
	linkedRecordName = the FQDN of the host record to which this service record is linked.	Both
	port = the TCP/UDP port on which the service is available.	Both

Object Type	Properties	Read-only/ Updatable
	priority = specifies which SRV record to use when multiple matching SRV records are present. The record with the lowest value takes precedence	Both
	weight = if two matching SRV records within a zone have equal priority, the weight value is checked. If the weight value for one object is higher than the other, the record with the highest weight has its resource records returned first.	Both
Text Record	ttl = time-to-live value	Both
	absolutName = the FQDN for the host record	Read-only
	txt	Both
Start of Authority Records	ttl = time-to-live value	Both

Admin

Object Type	Properties	Read-only/ Updatable
User	userType	Read-only
	securityPrivilege	Both
	historyPrivilege	Both
	email	Both
	phoneNumber	Both
	authenticator	Both
	userAccessType	Both
UserGroup	None	None
Authenticator	None	None

Tags

Object Type	Properties	Read-only/ Updatable
Tag	None	None
TagGroup	None	None

Vendor Profiles

Object Type	Properties	Read-only/ Updatable
VendorProfile	identifier = the Vendor Class Identifier	Both

Object Type	Properties	Read-only/ Updatable
VendorProfileOption	optionId = DNS Vendor Option ID	Read-only
	optionType = a data type for the option	Read-only
	optionDescription = a description of the information passed by the option	Both
	displayName = display name or screen name for the option.	Both
	optionAllowMultiple = allow the option to accept multiple values.	Read-only

DNSSEC

Object Type	Properties	Read-only/ Updatable
DNSSEC Signing Policies	None	None

TFTP Objects

Object Type	Properties	Read-only/ Updatable
TFTPGroup	None	None
TFTPFolder	None	None

MAC Pool Objects


Object Type	Properties	Read-only/ Updatable
MACAddress	address = String representing the mac address	Both
	macPool = Associated mac pool's name	Both
	macVendor = the IEEE MAC vendor of the MAC address. If not specified or found, Not Found is displayed.	Read-only
MACPool	None	None

Device

Object Type	Properties	Read-only/ Updatable
DeviceType	None	None
DeviceSubtype	None	None
Device	deviceTypeId = Id of associated DeviceType. If Device is associated with DeviceSubType, it will list the ID of the device type of the associated DeviceSubType.	Both

Object Type	Properties	Read-only/ Updatable
	deviceSubtypeId = Id of associated DeviceSubType (This property is available only if Device is associated with DeviceSubType.)	Both
	ip4Addresses = Comma delimited list of associated IP4Addresses.	Both
	ip6Addresses = Comma delimited list of associated IP6Addresses.	Both
TSIGKey	None	None

Location

Object Type	Properties	Read-only/ Updatable
Location	<p>code = The hierarchical location code consists of a set of 1 to 3 alpha-numeric strings separated by a space. The first two characters indicate a country, followed by next three characters which indicate a city in UN/LOCODE. New custom locations created under a UN/LOCODE city are appended to the end of the hierarchy. For example, CA TOR OF1 indicates:</p> <ul style="list-style-type: none"> • CA—Canada • TOR—Toronto • OF1—Office 1 <p> Note: The code is case-sensitive. It must be all UPPER CASE letters.</p> <p>The county code and child location code should be alphanumeric strings.</p>	Both
	country = The 2-digit country code. This property is only supported for get location methods.	Both
	description = The description of the location.	Both
	localizedname = The name of the location in your local language. For example, for Tokyo Office 1 in Japan, you can enter 東京オフィス1 in Japanese. If your location name contains diacritic marks, you can enter the name with diacritic marks in this field as well. For example, <i>Montréal</i> .	Both
	subDivision = The ISO 1-3 character alphabetic and/or numeric code for the administrative division of the country.	Both
	latitude = The geographical coordinate showing the north-south position of the location.	Both
	longitude = The geographical coordinate showing the east-west position of the location.	Both

Kerberos Realms


Object Type	Properties	Read-only/ Updatable
KerberosRealm	None	None
KDC	None	None
ServicePrincipal	None	None



Server

Object Type	Properties	Read-only/ Updatable
SingleServer	defaultInterfaceAddress = IP address of Server	Both
	fullHostName = Host name of Server	Both
	profile - Profile of server. Possible values are ADONIS_XMB2, ADONIS_XMB3, ADONIS_1950, ADONIS_1900, ADONIS_1200, ADONIS_800, DNS_DHCP_SERVER_20, DNS_DHCP_SERVER_45, DNS_DHCP_SERVER_60, DNS_DHCP_SERVER_100, DNS_DHCP_SERVER_100_D, WINDOWS_SERVER, OTHER_DNS_SERVER, PROTEUS_DDW, AFILIAS_DNS_SERVER.	Both
	activeNodeId = active server object ID passiveNodeId = passive server object ID activeNodePhysicalAddress = active server IP address passiveNodePhysicalAddress = passive server IP address Note: For xHA Server type only. If the server is not in xHA, this property is not available.	Read-only
	importViewName = name of the Windows view. Applicable only for Windows servers.	Both
	enableDNS = True if DNS is enabled, else false. This property will be present only if Server is a Windows server.	Both
	enableDHCP = True if DHCP is enabled, else false. This property will be present only if Server is windows server.	Both
	readOnly = True if Windows is added in read-only mode, else false. This property will be present only if Server is a Windows server.	Both
	authenticationCredentialUsername = Username for server. This property will be present only if Server is Windows and pmm server.	Both
	authenticationCredentialPassword - User password for server. This property will be updated only for Window server and PMM server.	Write-only
	authenticationCredentialDomain = Domain for server. This property will be present only if Server is Windows and pmm server.	Both
ScheduledDeployment	None	None

IPv4Objects



Object Type	Properties	Read-only/ Updatable
IP4Block	CIDR = CIDR value of the block. (if it forms a valid CIDR.)	Read-only
	name = name of the block	Both
	defaultDomains = Comma separated IDs of the default domains.	Both
	start = Start of the block. (if it does not form a valid CIDR)	Read-only
	end = End of the block. (if it does not form a valid CIDR)	Read-only
	defaultView = ID of the default View for the block.	Both
	dnsRestrictions = Comma separated IDs of the DNS zones or Views to restrict the IPv4 blocks to be used in.	Both
	allowDuplicateHost = Duplicate host names check option property. The possible values are Enable or Disable.	Both
	pingBeforeAssign = Ping check option property. The possible values are Enable or Disable.	Both
	inheritAllowDuplicateHost = Duplicate host names inheritance check option property. The possible values are True or False. If True, the AllowDuplicateHost option set at the parent object level will be used. If False, the allowDuplicateHost option must be specified and the value specified will be used.	Both
	inheritPingBeforeAssign = PingBeforeAssign option inheritance check option property. The possible values are True or False. If True, the PingBeforeAssign option set at the parent object level will be used. If False, the PingBeforeAssign option must be specified and the value specified will be used.	Both
	inheritDNSRestrictions = The possible values are True or False. If True, the IDs of the DNS zone or View to restrict the IPv4 blocks to be used in will be inherited from the parent object. If False, the DNSRestrictions option must be specified and the value specified will be used.	Both
	inheritDefaultDomains = The possible values are True or False. If True, the IDs of the default domain will be inherited from the parent object. If False, the DefaultDomains option must be specified and the value specified will be used.	Both
	inheritDefaultView = The possible values are True or False. If True, the ID of the default View for the block will be inherited from the parent object. If False, the DefaultView option must be specified and the value specified will be used.	Both
	locationCode = The hierarchical location code consists of a set of 1 to 3 alpha-numeric strings separated by a space. The first two characters indicate a country, followed by next three characters which indicate a city in UN/LOCODE. New custom locations created under a UN/LOCODE city are appended to the end of the hierarchy. For example, CA TOR OF1 indicates: CA= Canada TOR=Toronto OF1=Office 1.	Both

Object Type	Properties	Read-only/ Updatable
	 Note: The code is case-sensitive. It must be all UPPER CASE letters. The country code and child location code should be alphanumeric strings.	
	locationInherited = This defines if the location property was defined directly on the object level or was inherited from the parent object.	Read-only
IP4Network	CIDR = CIDR value of the block. (if it forms a valid CIDR.)	Read-only
	template = ID of the linked template.	Read-only
	gateway = Gateway of the network.	Both
	defaultDomains = Comma separated IDs of the default domains.	Both
	defaultView = ID of the default view for the block.	Both
	dnsRestrictions = Comma separated IDs of the DNS zones or views to restrict the IPv4 networks to be used in.	Both
	allowDuplicateHost = Duplicate host names check option property. The possible values are Enable or Disable.	Both
	pingBeforeAssign = Ping check option property. The possible values are Enable or Disable.	Both
	inheritAllowDuplicateHost = Duplicate host names inheritance check option property. The possible values are True or False. If True, the AllowDuplicateHost option set at the parent object level will be used. If False, the allowDuplicateHost option must be specified and the value specified will be used.	Both
	inheritPingBeforeAssign = PingBeforeAssign option inheritance check option property. The possible values are True or False. If True, the PingBeforeAssign option set at the parent object level will be used. If False, the PingBeforeAssign option must be specified and the value specified will be used.	Both
	inheritDNSRestrictions = The possible values are True or False. If True, the IDs of the DNS zone or View to restrict the IPv4 blocks to be used in will be inherited from the parent object. If False, the DNSRestrictions option must be specified and the value specified will be used.	Both
	inheritDefaultDomains = The possible values are True or False. If True, the IDs of the default domain will be inherited from the parent object. If False, the DefaultDomains option must be specified and the value specified will be used.	Both
	inheritDefaultView = The possible values are True or False. If True, the ID of the default View for the block will be inherited from the parent object. If False, the DefaultView option must be specified and the value specified will be used.	Both
	locationCode = The hierarchical location code consists of a set of 1 to 3 alpha-numeric strings separated by a space. The first two characters indicate a country, followed by next three characters which indicate a city in UN/LOCODE. New custom locations created under a UN/LOCODE city are appended to the end of the	Both

Object Type	Properties	Read-only/ Updatable
	<p>hierarchy. For example, CA TOR OF1 indicates: CA= Canada TOR=Toronto OF1=Office 1.</p> <p> Note: The code is case-sensitive. It must be all UPPER CASE letters. The country code and child location code should be alphanumeric strings.</p>	
	locationInherited = This defines if the location property was defined directly on the object level or was inherited from the parent object.	Read-only
IP4Address	address = Address string.	Read-only
	state = state of the address. For possible values, refer to IP Address States on page 252.	Read-only
	macAddress = MAC address of the IP4Address.	Both
	routerPortInfo = Connected router port information of the IPv4Address.	Read-only
	switchPortInfo = Connected switch port information of the IPvAddress.	Read-only
	vlanInfo = VLAN information of the IPv4Address.	Read-only
	<p>ptrs = a string providing unmanaged external host records with which the IPv4 address will be associated in the format:</p> <pre>viewId,exHostFQDN[, viedId,exHostFQDN,...]</pre>	Write-only
	leaseTime = the time when the IP address was leased.	Read-only
	expiryTime = the date and time that the DHCP lease expires. This is only for the DHCP Allocated IP address type.	Read-only
	parameterRequestList = the list of parameters the device requested from the DHCP server.	Read-only
	vendorClassIdentifier = an identifier sent by the DHCP client software running on a device.	Read-only
	<p>locationCode = The hierarchical location code consists of a set of 1 to 3 alpha-numeric strings separated by a space. The first two characters indicate a country, followed by next three characters which indicate a city in UN/LOCODE. New custom locations created under a UN/LOCODE city are appended to the end of the hierarchy. For example, CA TOR OF1 indicates: CA= Canada TOR=Toronto OF1=Office 1.</p> <p> Note: The code is case-sensitive. It must be all UPPER CASE letters. The country code and child location code should be alphanumeric strings.</p>	Both
	locationInherited = This defines if the location property was defined directly on the object level or was inherited from the parent object.	Read-only
IP4DHCPRange	start = Start of the range.	Both
	end = End of the range.	Both

Object Type	Properties	Read-only/ Updatable
	offset = IPv4 address from which the range should begin.	Both
	size = the size of the range to be created.	Both
	defineRangeBy = the possible values are OFFSET_AND_SIZE and OFFSET_AND_PERCENTAGE.	Both
IP4NetworkTemplate	gateway = gateway of the network.	Both
	reservedAddress = the list of reserved addresses being set on the network template.	Both

IPv6Objects

Object Type	Properties	Read-only/ Updatable
IP6Network	prefix = Prefix of the Network	Read-only
	locationCode = The hierarchical location code consists of a set of 1 to 3 alpha-numeric strings separated by a space. The first two characters indicate a country, followed by next three characters which indicate a city in UN/LOCODE. New custom locations created under a UN/LOCODE city are appended to the end of the hierarchy. For example, CA TOR OF1 indicates: CA= Canada TOR=Toronto OF1=Office 1.  Note: The code is case-sensitive. It must be all UPPER CASE letters. The country code and child location code should be alphanumeric strings.	Both
	locationInherited = This defines if the location property was defined directly on the object level or was inherited from the parent object.	Read-only
IP6Address	address = Address string	Read-only
	macAddress = MAC address of the IP6Address	Read-only
	state = State of the address. For possible values, refer to IP Address States on page 252.	Read-only
	ptrs = a string providing unmanaged external host records with which the IPv6 address will be associated in the format: <pre>viewId,exHostFQDN[, viedId,exHostFQDN,...]</pre>	Write-only
	locationCode = The hierarchical location code consists of a set of 1 to 3 alpha-numeric strings separated by a space. The first two characters indicate a country, followed by next three characters which indicate a city in UN/LOCODE. New custom locations created under a UN/LOCODE city are appended to the end of the hierarchy. For example, CA TOR OF1 indicates: CA= Canada TOR=Toronto OF1=Office 1.  Note: The code is case-sensitive. It must be all UPPER CASE letters. The country code and child location code should be alphanumeric strings.	Both

Object Type	Properties	Read-only/ Updatable
	locationInherited = This defines if the location property was defined directly on the object level or was inherited from the parent object.	Read-only
IP6DHCPRange	start = Start of the range	Read-only
	end = End of the range	Read-only
	size = the size of the range to be created.	Read-only
	defineRangeBy = the possible values are AUTOCREATE_BY_SIZE, OFFSET_AND_SIZE, and START_ADDRESS_AND_SIZE	Read-only

DeploymentRoles

Object Type	Properties	Read-only/ Updatable
DNSDeploymentRole	view	Both
	zoneTransServerInterface	Both
	inherited	Read-only
DHCPDeploymentRole	inherited	Read-only
	secondaryServerInterfaceId	Both

Access right

Object Type	Properties	Read-only/ Updatable
Access right	workflowLevel = valid values are None, Recommend, and Approve.	Both
	deploymentAllowed = true to perform a full deployment of data to a managed server, else false.	Both
	quickDeploymentAllowed = ture to instantly deploy changed DNS resource records, else false.	Both

IP Address States

The following tables list the available values of the state parameter of the IP address.

IPv4

State	Description
UNALLOCATED	Available and unassigned IP address for DNS or DHCP.
STATIC	Statically assigned hosts and only used for DNS purposes.
DHCP_ALLOCATED	Dynamically assigned through DHCP to the given MAC address.
DHCP_FREE	Dynamically assigned through DHCP but are now in a free or unallocated state.

State	Description
DHCP_RESERVED	Represent DHCP reservations, and may yet be assigned to a host. These addresses can be inside or outside of a DHCP range.
DHCP_LEASED	Used in a DHCP lease.
RESERVED	Reserved for future use. While reserved, the address cannot be assigned a DNS host name and cannot be deployed to DHCP.
GATEWAY	Network gateway (router) addresses.

IPv6

State	Description
UNALLOCATED	Available and unassigned IP address for DNS or DHCP.
STATIC	Statically assigned hosts and only used for DNS purposes.
DHCP_ALLOCATED	Dynamically assigned through DHCP to the given MAC address.
DHCP_FREE	Dynamically assigned through DHCP but are now in a free or unallocated state.
DHCP_RESERVED	Represent DHCP reservations, and may yet be assigned to a host. These addresses can be inside or outside of a DHCP range.
DHCP_LEASED	Used in a DHCP lease.
RESERVED	Reserved for future use. While reserved, the address cannot be assigned a DNS host name and cannot be deployed to DHCP.
GATEWAY	Network gateway (router) addresses.

Index

A

Address Manager API

API Objects

APIEntity Class [25](#)

Java API Examples

Available Java Classes [33](#)

Connecting [28](#)

Logging in [29](#)

Perl API Examples

Adding [35](#)

Connecting [34](#)

Deleting [35](#)

Getting [35](#)

Logging in [34](#)

Updating [35](#)

REST API [35](#)

SOAP ports [24](#)

Web Services API

SOAP Binding Address [24](#)

API Constants

Access Right Values [189](#)

Additional IP Service Type [189](#)

Configuration Setting [189](#)

Defined probe values [217](#)

Deployment Services [189](#)

Deployment Status [189](#)

Device Properties [190](#)

DHCP6 Client Options [194](#)

DHCP6 Service Options [196](#)

DHCP Class Match Criteria [191](#)

DHCP Client Options [191](#)

DHCP Custom Option Types [194](#)

DHCP Deployment Role Types [195](#)

DHCPServiceOptionConstants [196](#)

DHCP Service Options [195](#)

DNS Deployment Role Type [197](#)

DNS Options [197](#), [198](#)

DNSSEC Key Format [199](#)

DNS Zones Deployment Validation Check [199](#)

Entity Categories [199](#)

ENUM Services [200](#)

IP Assignment Action Values [201](#)

IP discovery type [201](#)

Object Properties [201](#)

Object Types [209](#)

Option Types [211](#)

PositionRangeBy [212](#)

Probe status values [217](#)

Response Policy Search [212](#)

Response Policy Type [212](#)

Reverse Zone Format Type [212](#)

Server Capability Profiles [213](#)

Service Types [213](#)

SNMP authentication type [214](#)

SNMP Privacy type [214](#)

SNMPSecurityLevels [214](#)

SNMP Version [213](#)

Traversal Methodology [214](#)

User Access Type [215](#)

User-defined Field Type [215](#)

User-defined Field Validator Properties [215](#)

User History Privileges [215](#)

User Security Privileges [216](#)

User Type [216](#)

Vendor Profile Option Types [216](#)

Workflow Levels [217](#)

API Method Reference

Address Manager Objects

Access Rights [237](#)

Authenticators [237](#)

Collecting Data [240](#)

Configurations [236](#)

Database Management [239](#)

Devices [238](#)

Groups and Users [237](#)

Location [238](#)

MAC Addresses [239](#)

MAC Pools [239](#)

Migration [240](#)

Object Tag Groups [238](#)

Object Tags [238](#)

Workflow Change Requests [239](#)

API Sessions [220](#)

Crossover High Availability (XHA) [236](#)

Deployment Options [233](#)

DHCP

DHCP6 Client Options [226](#)

DHCP6 Service Options [227](#)

DHCP Client Options [226](#)

DHCP Custom Options [227](#)

DHCP Match Classes [228](#)

DHCP Service Options [227](#)

DHCP Vendor Options [227](#)

IPv4 DHCP Ranges [225](#)

IPv6 DHCP Ranges [226](#)

Shared Networks [228](#)

DNS

Alias Records [231](#)

DNS Options [232](#)

DNS Response Policies [233](#)

DNS Views [228](#)

DNS Zones [229](#)

DNS Zone Templates [229](#)

ENUM Numbers [229](#)

ENUM Zones [229](#)

External Host Records [230](#)

Generic Records [232](#)

Generic Resource Records [230](#)

HINFO Records [231](#)

Host Records [230](#)

MX Records [231](#)

NAPTR Records [230](#)

Reverse zone name format [233](#)

SRV Records [232](#)

Start of Authority Records [232](#)

- Text Records [231](#)
- Generic Methods
 - Changing Locale [221](#)
 - Liked Entities [220](#)
- IPAM
 - IP Discovery and Reconciliation [224](#)
 - IPv4 Addresses [223](#)
 - IPv4 Blocks [221](#)
 - IPv4 Group [224](#)
 - IPv4 Networks [222](#)
 - IPv4 Network Templates [223](#)
 - IPv4 Objects [223](#)
 - IPv6 Objects [224](#)
 - Provision Devices [225](#)
- Servers and Deployment
 - DHCP Deployment Roles [235](#)
 - DNS and DHCP Deployment Roles [235](#)
 - DNS Deployment Roles [236](#)
 - Server Group [235](#)
 - Servers [234](#)
 - TFTP Deployment Roles [236](#)
- TFTP
 - TFTP Files [234](#)
 - TFTP Folders [234](#)
 - TFTP Groups [233](#)
- User-defined Fields [221](#)
- API Object Methods
 - Address Manager Objects
 - Access Rights
 - Add Access Right [170](#)
 - Delete Access Rights [173](#)
 - Get Access Right [171](#)
 - Get Access Rights for Entity [171](#)
 - Get Access Rights for User [172](#)
 - Update Access Rights [172](#)
 - Authenticators
 - Authenticator Generic Methods [170](#)
 - Update Authenticator [170](#)
 - Configurations
 - Add Configuration [165](#)
 - Configuration Generic Methods [166](#)
 - Get Configuration Setting [166](#)
 - Update Configuration [166](#)
 - Update Configuration Setting [166](#)
 - Database Management
 - Configure Replication [177](#)
 - Purge history [177](#)
 - Devices
 - Add Device [178](#)
 - Add Device Subtype [179](#)
 - Add Device Type [179](#)
 - Groups and Users
 - Add Group [167](#)
 - Add User [167](#)
 - Group Generic Methods [167](#)
 - Update Group [167](#)
 - update password [169](#)
 - Update User [169](#)
 - User Generic Methods [169](#)
 - Location Groups [175](#)
 - Locations
 - Add Location [175](#)
 - Get Location By Code [176](#)
 - Get Locations [176](#)
 - MAC Addresses
 - Add MAC Address [181](#)
 - Associate MAC Address [181](#)
 - Deny MAC Address [181](#)
 - Is Address Allocated? [182](#)
 - MAC Address Generic Methods [182](#)
 - Update MAC Address [182](#)
 - MAC Pools
 - Add MAC Pool [180](#)
 - Get MAC Addresses in Pool [180](#)
 - MAC Pool Generic Methods [180](#)
 - Update MAC Pool [180](#)
 - Object Tag Groups
 - Add Object Tag Group [173](#)
 - Object Tag Group Generic Methods [174](#)
 - Update Object Tag Group [174](#)
 - Object Tags
 - Add Object Tag [174](#)
 - Assign Object Tag [174](#)
 - Object Tag Generic Methods [175](#)
 - Remove Object Tag [175](#)
 - Update Object Tag [175](#)
 - Workflow Change Requests [182](#)
 - Changing Locale
 - Log in with Options [55](#)
 - Crossover High Availability (xHA)
 - Breaking an xHA
 - Break xHA [164](#)
 - Creating an xHA
 - Create xHA [161](#)
 - Edit xHA [162](#)
 - Requirements for creating an xHA pair [161](#)
 - xHA Failover
 - Failover xHA [165](#)
 - Data collection
 - Get probe data [184](#)
 - Get probe status [184](#)
 - Start probe [184](#)
 - Deployment options
 - Add Raw Deployment Option [144](#)
 - Get Deployment Options [143](#)
 - Getting deployment options [143](#)
 - Raw deployment options [144](#)
 - Update Raw Deployment Option [145](#)
 - DHCP
 - DHCP6 Client Options [100](#)
 - DHCP6 Service Options [105](#)
 - DHCP Client Options [99](#)
 - DHCP Custom Options [102](#)
 - DHCP Match Classes
 - Add DHCP Match Classes [110](#)
 - Add DHCP Sub Classes [111](#)
 - Delete DHCP Match Classes [111](#)
 - Delete DHCP Sub Classes [112](#)
 - Update DHCP Match Classes [111](#)
 - Update DHCP Sub Classes [111](#)
 - DHCP Raw Options [112](#)
 - DHCP Service Options [103](#)
 - DHCP Vendor Profiles and Options [106](#)
 - IPv4 DHCP Ranges [93](#)

- IPv6 DHCP Ranges [96](#)
- Shared Networks
 - Get Shared Networks [113](#)
 - Link a Shared Network Tag [113](#)
 - Unlink a Shared Network Tag [113](#)
- DNS
 - DNS Options
 - Add DNS Option [137](#)
 - Delete DNS Option [138](#)
 - DNS Option value formats [138](#)
 - Get DNS Option [137](#)
 - Update DNS Option [138](#)
 - DNS Raw Option [139](#)
 - DNS Resource Records
 - Alias Records
 - Add Alias Record [130](#)
 - Alias Record Generic Methods [131](#)
 - Get Aliases by Hint [130](#)
 - Update Alias Record [131](#)
 - Dotted resource records [122](#)
 - External Host Records
 - Add External Host Record [125](#)
 - External Host Record Generic Methods [126](#)
 - Get IP address with External Host records [126](#)
 - Update External Host Record [126](#)
 - Generic Records
 - Add Generic Record [136](#)
 - Generic Record Generic Methods [136](#)
 - Update Generic Record [136](#)
 - Generic Resource Records
 - Add Entity for Resource Records [124](#)
 - Add Resource Records [123](#)
 - Move Resource Records [124](#)
 - HINFO Records
 - Add HINFO Record [132](#)
 - HINFO Record Generic Methods [132](#)
 - Update HINFO Record [132](#)
 - Update MX Record [133](#)
 - Host Records
 - Add Bulk Host Records [127](#)
 - Add Host Record [127](#)
 - Get Dependent Records [129](#)
 - Get Host Record by Hint [128](#)
 - Get IP Address with Host Records [129](#)
 - Host Record Generic Methods [129](#)
 - Update Host Record [129](#)
 - MX Records
 - Add MX Record [133](#)
 - MX Record Generic Methods [133](#)
 - NAPTR Records
 - Add NAPTR Record [124](#)
 - NAPTR Record Generic Methods [125](#)
 - Update NAPTR Record [125](#)
 - SRV Records
 - Add SRV Record [134](#)
 - SRV Record Generic Methods [134](#)
 - Update SRV Record [134](#)
 - Start of Authority Records
 - Add Start of Authority Records [134](#)
 - Start of Authority Record Generic Methods [135](#)
 - Update Start of Authority Record [135](#)
 - Text Records
 - Add Text Record [131](#)
 - Text Record Generic Methods [132](#)
 - Update Text Record [132](#)
 - DNS Response Policies
 - Add Response Policy [140](#)
 - Response Policy Generic Methods [141](#)
 - Search Response Policy [141](#)
 - Update Response Policy [140](#)
 - Upload Response Policy Item [141](#)
 - ENUM Numbers
 - Add ENUM Number [121](#)
 - ENUM Number Generic Methods [121](#)
 - Update ENUM Number [121](#)
 - ENUM Zones
 - Add ENUM Zone [120](#)
 - ENUM Zone Generic Methods [121](#)
 - Update ENUM Zone [121](#)
 - Reverse zone name format
 - Add Reverse Zone Name Format [142](#)
 - Zones
 - Add Entity for DNS Zones [115](#)
 - Add Zone [116](#)
 - Get Key Signing Key [117](#)
 - Get Zones by Hint [116](#)
 - Update Zone [117](#)
 - Zone Generic Methods [82](#), [117](#)
 - Zone templates
 - Add Records to DNS Zone Template [120](#)
 - Add Zone Template [118](#)
 - Assign or Update Template [118](#)
 - Update Zone Template [120](#)
 - Zone Template Generic Methods [120](#)
- DNS Views
 - Add Access Control List (ACL) [115](#)
 - Add DNS View [114](#)
 - DNS View Generic Methods [115](#)
 - Update Access Control List (ACL) [115](#)
 - Update DNS View [114](#)
- Generic Methods
 - Deleting Objects
 - Delete [52](#)
 - Delete with Options [52](#)
 - getting objects
 - Get Entities [44](#)
 - Get Entity by ID [44](#)
 - Get Entity by Name [44](#)
 - Get Parent [45](#)
 - Linked Entities
 - Get Linked Entities [53](#)
 - Link Entities [53](#)
 - Unlink Entities [54](#)
 - search and retrieve entities
 - Custom search [46](#)
 - Get Entities by Name [49](#)
 - Get Entities by Name Using Options [49](#)
 - Get MAC Address [50](#)
 - Search by Category [48](#)
 - Search by Object Types [48](#)

- update objects
 - examples [51](#)
 - Update [50](#), [51](#)
 - Update with Options [51](#)
 - IPAM
 - Additional IP Addresses [76](#)
 - IP Discovery and Reconciliation
 - Add IPv4 Reconciliation Policy [80](#)
 - Get Discovered Device [83](#)
 - Get Discovered Device ARP Entries [84](#)
 - Get Discovered Device Hosts [84](#)
 - Get Discovered Device Interfaces [83](#)
 - Get Discovered Device MAC Address Entries [85](#)
 - Get Discovered Device Networks [83](#)
 - Get Discovered Devices [83](#)
 - Get Discovered Device Vlans [84](#)
 - IPv4 addresses [71](#)
 - IPv4 Blocks
 - Add IPv4 Block by CIDR [58](#)
 - Add IPv4 Block by Range [58](#)
 - Add Parent Block [59](#)
 - IPv4 Discovery and Reconciliation [80](#)
 - IPv4 Group [77](#)
 - IPv4 networks
 - Add IPv4 Network [62](#)
 - IPv4 network templates [67](#)
 - IPv4 objects [78](#)
 - IPv6 objects [85](#)
 - Provision devices [91](#)
 - Migration
 - Migrate a File [183](#)
 - Migration Status [183](#)
 - Servers and Deployment
 - DNS and DHCP Deployment Roles
 - DHCP Deployment Roles
 - Add DHCP Deployment Role [156](#)
 - Delete DHCP Deployment Role [157](#)
 - Get DHCP Deployment Role [156](#)
 - Update DHCP Deployment Role [157](#)
 - DNS Deployment Roles
 - Add DNS Deployment Role [158](#)
 - Delete DNS Deployment Role [159](#)
 - Delete DNS Deployment Role for View [159](#)
 - Get DNS Deployment Role [158](#)
 - Get DNS Deployment Role for View [159](#)
 - Update DNS Deployment Role [159](#)
 - Get Deployment Roles for DNS and IP Address Space Objects [155](#)
 - Get Servers Associated with a Deployment Role [154](#)
 - Get Server's Associated Deployment Roles [155](#)
 - Move Deployment Roles [155](#)
 - TFTP Deployment Roles
 - Add TFTP Deployment Role [160](#)
 - TFTP Deployment Role Generic Methods [160](#)
 - Update TFTP Deployment Role [160](#)
 - Server Group
 - Add Server Group [154](#)
 - Generic methods [154](#)
 - Link server to server group [154](#)
 - Unlink server from server group [154](#)
 - update [154](#)
- Servers
 - Add Server [148](#)
 - Deployment Status [153](#)
 - Deploy Server [151](#)
 - Deploy Server Configuration [151](#)
 - Deploy Server Services [152](#)
 - Get Published Interface [153](#)
 - Import Server [149](#)
 - Quick Deployment [152](#)
 - Replace Server [150](#)
 - Server Generic Methods [153](#)
- TFTP
 - TFTP Files
 - Add TFTP File [146](#)
 - TFTP File Generic Methods [147](#)
 - Update TFTP File [147](#)
 - TFTP Folders
 - Add TFTP Folder [146](#)
 - TFTP Folder Generic Methods [146](#)
 - Update TFTP Folder [146](#)
 - TFTP Groups
 - Add TFTP Group [145](#)
 - TFTP Group Generic Methods [145](#)
 - Update TFTP Group [145](#)
- User-defined Fields
 - Getting User-defined fields
 - Get User-defined Field [57](#)
 - Update Bulk User-defined Field [57](#)
 - Java API Examples [56](#)
 - Perl API Examples [56](#)
- API Objects
 - APIAccessRight Class [25](#)
 - APIDeploymentOption Class [26](#)
 - APIDeploymentRole Class [25](#)
 - APIUserDefinedField Class [26](#)
 - ResponsePolicySearchResult Class [27](#)
- API Sessions
 - connect to Address Manager [27](#)
- ## D
- delete objects [31](#)
 - DHCP
 - DHCP6 Client Options
 - Add DHCP6 Client Option [100](#)
 - Delete DHCP6 Client Option [101](#)
 - Get DHCP6 Client Option [101](#)
 - Update DHCP6 Client Option [101](#)
 - DHCP6 Service Options
 - Add DHCP6 Service Option [105](#)
 - Delete DHCP6 Service Option [106](#)
 - Get DHCP6 Service Option [105](#)
 - Update DHCP6 Service Option [106](#)
 - DHCP Client Options
 - Add DHCP Client Option [99](#)
 - Delete DHCP Client Option [100](#)
 - Get DHCP Client Option [99](#)
 - Update DHCP Client Option [100](#)

- DHCP Custom Options
 - Add Custom Deployment Option [102](#)
- DHCP Service Options
 - Add DHCP Service Option [103](#)
 - Delete DHCP Service Option [104](#)
 - Get DHCP Service Option [104](#)
 - Update DHCP Service Option [104](#)
- DHCP Vendor Profiles and Options
 - Add DHCP Vendor Deployment Option [107](#)
 - Add Vendor Option Definition [107](#)
 - Add Vendor Profile [108](#)
 - Delete DHCP Vendor Deployment Option [108](#)
 - Get DHCP Vendor Deployment Option [109](#)
 - Update DHCP Vendor Deployment Option [109](#)
- Get IPv4 DHCP Ranges [95](#)
- Get Max Allowed Range [96](#)
- IPv4 DHCP Range Generic Methods [96](#)
- IPv6 DHCP Ranges
 - Add IPv6 DHCP Range [96](#)
 - Get IPv6 DHCP Range [98](#)
 - Get IPv6 Range by IP Address [97](#)
 - Get Multiple IPv6 DHCP Ranges [98](#)
 - IPv6 DHCP Range Generic Methods [99](#)
 - Update IPv6 DHCP Range [99](#)
- Update IPv4 DHCP Range [96](#)

I

IPAM

- Additional IP Addresses
 - Add Additional IP Addresses [76](#)
 - Get Additional IP Addresses [77](#)
 - Remove Additional IP Addresses [76](#)
- Delete Device Instance [93](#)
- DHCP
 - Add IPv4 DHCP Range [94](#)
 - Add IPv4 DHCP Range by size [94](#), [97](#)
 - Get IPv4 DHCP Range [95](#)
 - Get IPv4 Range by IP Address [94](#)
- IPv4 addresses
 - assign IPv4 addresses [71](#)
 - Assign Next Available IPv4 Address [72](#)
 - Change IPv4 Address State [75](#)
 - Check Allocation for IPv4 Address [74](#)
 - Get IPv4 Address [73](#)
 - Get Next Available Address [75](#)
 - Get Next IPv4 Address [74](#)
 - IPv4 Address Generic Methods [75](#)
 - Update IPv4 Address [75](#)
- IPv4 Blocks
 - add parent block with properties [59](#)
 - get IP range by IP address [59](#)
 - get IPv4 Block by CIDR [60](#)
 - get IPv4 Block by Range [60](#)
 - Merge blocks with parent [61](#)
 - Merge selected blocks or networks [61](#), [61](#), [61](#)
- IPv4 Group
 - Add IPv4 IP Group by Range [77](#)
 - Add IPv4 IP Group by Size [78](#)
- IPv4 Networks
 - Add IPv4 Network Template [67](#), [68](#)
 - Get IPv4 network by CIDR [62](#)

- Get IPv4 Network by Hint [63](#)
- Get IPv4 Network by Range [64](#)
- Get IPv4 Range by IP Address [62](#)
- Get Next Available IP Range [65](#)
- Get Next Available IP Ranges [65](#)
- Get Next Available Network [64](#)
- IPv4 Network Generic Methods [67](#)
- IPv4 Network Template Generic Methods [71](#)
- Re-apply Template [69](#)
- Split IPv4 Network [66](#)
- Update IPv4 Network [67](#)
- Update IPv4 Network Template Name [70](#)
- IPv4 objects
 - Move IP object [79](#)
 - Move IPv4 object [78](#)
 - Resize Range [79](#)
- IPv6 objects
 - Add Device Instance [91](#)
 - Add IPv6 Address [85](#)
 - Add IPv6 Block by MAC Address [86](#)
 - Add IPv6 Block by Prefix [86](#)
 - Add IPv6 Network by Prefix [86](#)
 - Assign IPv6 Address [88](#)
 - Clear IPv6 Address [90](#)
 - Get Entity by Prefix [90](#)
 - Get IPv6 Address [90](#)
 - Get IPv6 objects by Hint [87](#)
 - Get IPv6 Range by IP Address [87](#)
 - Reassign IPv6 Address [91](#)
 - Split IPv6 Block [87](#)
 - Split IPv6 Network [87](#)
- Provision devices [93](#)

O

- Objects [18](#)

P

Property Options Reference

- IP Address States [252](#)
- Property options
 - Access right [252](#)
 - Admin [244](#)
 - Configuration [242](#)
 - DeploymentRoles [252](#)
 - Device [245](#)
 - DNSSEC [245](#)
 - IPv4 [252](#)
 - IPv4Objects [248](#)
 - IPv6 [253](#)
 - IPv6Objects [251](#)
 - Kerberos Realms [247](#)
 - Location [246](#)
 - MAC Pool Objects [245](#)
 - Resource Records [242](#)
 - Server [247](#)
 - Tags [244](#)
 - TFTP Objects [245](#)
 - Vendor Profiles [244](#)
 - Views and Zones [242](#)

S

SOAP Binding Address [24](#)

SSL [21](#)

W

What's New [13](#)



BlueCat Networks (USA) Inc. and its affiliates.

www.bluecatnetworks.com

Toll Free: 1.866.895.6931

Document #: BAM_API_v8.3.0-R1

Published in Canada

Date: November 2017