Cisco UCS Central XML API Programmer's Guide

First Published: July 23, 2014

Last Modified: August 01, 2016

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Preface

Audience

This guide is intended primarily for data center administrators with responsibilities and expertise in one or more of the following:

- Server administration
- Storage administration
- Network administration
- Network security

Conventions

Text Type	Indication
GUI elements	GUI elements such as tab titles, area names, and field labels appear in **this font**. Main titles such as window, dialog box, and wizard titles appear in **this font**.
Document titles	Document titles appear in *this font*.
TUI elements	In a Text-based User Interface, text the system displays appears in `this font`.
System output	Terminal sessions and information that the system displays appear in `this font`.

Text Type	Indication
CLI commands	CLI command keywords appear in **this font**. Variables in a CLI command appear in *this font*.
[ ]	Elements in square brackets are optional.
{x | y | z}	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
< >	Nonprinting characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the document.

Tip Means *the following information will help you solve a problem*. The tips information might not be troubleshooting or even an action, but could be useful information, similar to a Timesaver.

Timesaver Means *the described action saves time*. You can save time by performing the action described in the paragraph.

Caution Means *reader be careful*. In this situation, you might perform an action that could result in equipment damage or loss of data.

Warning IMPORTANT SAFETY INSTRUCTIONS

This warning symbol means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Use the statement number provided at the end of each warning to locate its translation in the translated safety warnings that accompanied this device.

SAVE THESE INSTRUCTIONS

Related Cisco UCS Documentation

Documentation Roadmaps

For a complete list of all B-Series documentation, see the *Cisco UCS B-Series Servers Documentation Roadmap* available at the following URL: http://www.cisco.com/go/unifiedcomputing/b-series-doc.

For a complete list of all C-Series documentation, see the *Cisco UCS C-Series Servers Documentation Roadmap* available at the following URL: http://www.cisco.com/go/unifiedcomputing/c-series-doc.

For information on supported firmware versions and supported UCS Manager versions for the rack servers that are integrated with the UCS Manager for management, refer to Release Bundle Contents for Cisco UCS Software.

Other Documentation Resources

Follow Cisco UCS Docs on Twitter to receive document update notifications.

Documentation Feedback

To provide technical feedback on this document, or to report an error or omission, please send your comments to ucs-docfeedback@cisco.com. We appreciate your feedback.

CHAPTER 1

Cisco UCS Central XML API

This chapter includes the following sections:

Cisco UCS Central Overview

Cisco UCS Central provides scalable management solutions for growing Cisco UCS environment. Cisco UCS Central simplifies the management of multiple Cisco UCS domains from a single management point through standardization, global policies and global ID pools. Cisco UCS Central does not replace Cisco UCS Manager, which is the policy driven management for single UCS domain. Instead Cisco UCS Central focuses on managing and monitoring the UCS domains on a global level, across multiple individual Cisco UCS Classic and Mini management domains worldwide.

Cisco UCS Central enables you to manage individual or groups of Cisco UCS domains with the following:

- Centralized Inventory of all Cisco UCS components for a definitive view of the entire infrastructure and simplified integration with current Information Technology Infrastructure Library (ITIL) processes.
- Centralized, policy-based firmware upgrades that can be applied globally or selectively through automated schedules or as business workloads demand
- Global ID pooling to eliminate identifier conflicts
- Global administrative policies that enable both global and local management of the Cisco UCS domains
- An XML API, building on the Cisco UCS Manager XML API for easy integration into higher-level data center management frameworks
- Bandwidth statistics collection and aggregation with two week or one year retention

- Remote management to manage various end points in registered Cisco UCS domains

Cisco UCS Central does not reduce or change any local management capabilities of Cisco UCS Manager, such as its API. This allows you to continue usingCisco UCS Manager the same way as when you did not have Cisco UCS Central, and also allows all existing third party integrations to continue to operate without change.

Cisco UCS Central Features

The following table provides a list of features with brief description on the management capabilities of Cisco UCS Central:

Feature	Description
Centralized inventory	Cisco UCS Central automatically aggregates a global inventory of all registered Cisco UCS components, organized by domain, with customizable refresh schedules and provides even easier integration with ITIL processes, with direct access to the inventory through an XML interface.
Centralized fault summary	Cisco UCS Central enables you to view the status of all Cisco UCS infrastructure on the global fault summary panel, with a fault summary organized by domain and fault type. Also provides you the ability to view individual Cisco UCS Manager domains for greater fault detail and more rapid problem resolution. Drilling down on a fault launches the UCS Manager in context for a seamlessly integrated experience.
Centralized, policy-based firmware upgrades	You can download firmware updates automatically from the Cisco.com to a firmware library within Cisco UCS Central. Then schedule automated firmware updates, globally or selectively, based on your business requirements. Managing firmware centrally ensures compliance with IT standards and makes reprovisioning of resources a point-and-click operation.
Global ID pools	Cisco UCS Central eliminates identifier conflicts and ensures portability of software licenses. You are able to centralize the sourcing of all IDs, such as universal user IDs (UUIDs), MAC addresses, IP addresses, and worldwide names (WWNs), from global pools and gain real-time ID use summaries. Centralizing server identifier information makes it simple to move a server identifier between Cisco UCS domains anywhere in the world and reboot an existing workload to run on the new server.

Feature	Description
Domain groups	Cisco UCS Central simplifies policy management by providing options to create domain groups and subgroups. A domain group is an arbitrary grouping of Cisco UCS domains that can be used to group systems into geographical or organizational groups. Each domain group can have up to five levels of domain sub groups. This provides you the ability to manage policy exceptions when administering large numbers of Cisco UCS domains. Each sub group has a hierarchical relationship with the parent domain group.
Global administrative policies	Cisco UCS Central helps you to ensure compliance and staff efficiency with global administrative policies. The global policies are defined at the domain group level and can manage anything in the infrastructure, from date and time and user authentication to equipment power and system event log (SEL) policies.
Global service profiles and templates	Global service profiles and templates in Cisco UCS Central enables fast and simplified infrastructure deployment and provides consistency of configurations throughout the enterprise. This feature enables global bare-metal workload mobility very similar to how hypervisor enables virtualized workload mobility.
Statistics management	Cisco UCS Central enables you to gain a better understanding of how Cisco UCS domains are functioning over time to improve operations to smoothly handle periodic peaks and shifts in workload. You can configure and generate reports from the Cisco UCS Central GUI. To accelerate the collection of statistics, the centralized database schema is open and data can be accessed directly or through the Cisco UCS Central Software GUI, command-line interface (CLI), or XML API.
Backup	Cisco UCS Central provides an automatic backup facility that enables quick and efficient backing up the configuration information of the registered Cisco UCS domains and the UCS Central configuration.
High availability	As with all Cisco UCS solutions, Cisco UCS Central is designed for no single point of failure. High availability for Cisco UCS Central Software allows organizations to run Cisco UCS Central using an active-standby model with a heartbeat that automatically fails over if the active Cisco UCS Central does not respond.
XML API	Cisco UCS Central, just like Cisco UCS Manager, has a high-level industry-standard XML API for interfacing with existing management frameworks and orchestration tools. The XML API for Cisco UCS Central Software is similar to the XML API for Cisco UCS Manager, making integration with high-level managers very fast.

Feature	Description
Remote Management	Cisco UCS Central enables you to manage various end points in the registered Cisco UCS domains from one management point. You can manage chassis, servers, fabric interconnects, and fabric extenders from Cisco UCS Central GUI or CLI. You can also access tech support files for registered UCS domains from Cisco UCS Central.
Policy/policy component and resources import	Cisco UCS Central provides you the flexibility search for and import a perfect policy/policy component or a resource from one registered UCS domain into CCisco UCS Central. You can then deploy this policy or the resource to other managed domains.

Domain Groups

Cisco UCS Central creates a hierarchy of Cisco UCS domain groups for managing multiple Cisco UCS domains. You will have the following categories of domain groups in Cisco UCS Central:

- **Domain Group—** A group that contains multiple Cisco UCS domains. You can group similar Cisco UCS domains under one domain group for simpler management.
- **Ungrouped Domains—**When a new Cisco UCS domain is registered in Cisco UCS Central, it is added to the ungrouped domains. You can assign the ungrouped domain to any domain group.

If you have created a domain group policy, and a new registered Cisco UCS domain meets the qualifiers defined in the policy, it will automatically be placed under the domain group specified in the policy. If not, it will be placed in the ungrouped domains category. You can assign this ungrouped domain to a domain group.

Each Cisco UCS domain can only be assigned to one domain group. You can assign or reassign membership of the Cisco UCS domains at any time. When you assign a Cisco UCS domain to a domain group, the Cisco UCS domain will automatically inherit all management policies specified for the domain group.

Before adding a Cisco UCS domain to a domain group, make sure to change the policy resolution controls to local in the Cisco UCS domain. This will avoid accidentally overwriting service profiles and maintenance policies specific to that Cisco UCS domain. Even when you have enabled auto discovery for the Cisco UCS domains, enabling local policy resolution will protect the Cisco UCS domain from accidentally overwriting policies.

Policies

Cisco UCS Central acts as a global policy server for registered Cisco UCS domains. Configuring global Cisco UCS Central policies for remote Cisco UCS domains involves registering domains and assigning registered domains to domain groups.

In addition, the policy import capability allows a local policy to be globalized inside of Cisco UCS Central. You can then apply these global policies to other registered Cisco UCS domains.

You can define global policies in Cisco UCS Central that are resolved by Cisco UCS Manager in a registered Cisco UCS domain.

Policy Browser

The policy browser contains information about policies from registered UCS domains and provides meaningful co-related information from user queries. This information may be used to import policies from those participating UCS domains.

The following features are supported:

- Simplify management of multiple policies across UCS domains by providing a centralized view.
- While administrators can define the policies in individual UCS domains, policies can be imported into Cisco UCS Central from those domains.
- Provides seamless views for pulling information in the simplest form about policies, based on locality, type or policy.
- Greater level of administrative control to import a robust UCS management domain's policy built-up into Cisco UCS Central for ease of migration.

The policy browser structure is intended for administrators of multiple UCS management domains, and to help those administrators in bringing the policy symmetry and providing policy repository for simplification of operations. Users with administrative privileges can use the policy browser.

The policy browser manage policy data in UCS management domains and use basic APIs that are externally visible for other north bound API clients, to share this data. Policy browser information can be extracted using these APIs.

Cisco UCS Central maintains the information about policies and resources in the resource manager along with MIT objects. These objects are updated after reading the UCSM MIT objects through the inventory. A consolidated tree is maintained in Cisco UCS Central for further queries from the GUI and north bound XML APIs to retrieve remote policy information residing in UCS management domains.

Pools

Pools are collections of identities, or physical or logical resources, that are available in the system. All pools increase the flexibility of service profiles and allow you to centrally manage your system resources. Pools that are defined in Cisco UCS Central are called **Global Pools** and can be shared between Cisco UCS domains. **Global Pools** allow centralized ID management across Cisco UCS domains that are registered with Cisco UCS Central. By allocating ID pools from Cisco UCS Central to Cisco UCS Manager, you can track how and where the IDs are used, prevent conflicts, and be notified if a conflict occurs. Pools that are defined locally in Cisco UCS Manager are called **Domain Pools**.

Note The same ID can exist in different pools, but can be assigned only once. Two blocks in the same pool cannot have the same ID.

You can pool identifying information, such as MAC addresses, to preassign ranges for servers that host specific applications. For example, you can configure all database servers across Cisco UCS domains within the same range of MAC addresses, UUIDs, and WWNs.

Information Sharing between UCS Domains and Cisco UCS Central

When you log on to UCS Manager for the first time, it performs a quick scan on all the policies to build and initialize the UCS local policy map. Subsequently policy and resource objects are created, updated, and deleted.

Once the local objects created and built, these are reported to Cisco UCS Central using the inventory update mechanism. Upon receiving these objects, the policy:Universe object gets updated and the element objects are transformed into cluster and source objects. These objects are updated and deleted on the UCS domains with corresponding state changes.

When you unregister the UCS domain from Cisco UCS Central, the policy sources are removed for those UCS domains from the clusters. If no source objects exist under a cluster, that particular cluster is released.

Introduction to the Cisco UCS Central XML API

Cisco UCS Central provides you the ability to manage multiple Cisco UCS domains with different versions of Cisco UCS Manager at the same time. Cisco UCS Central identifies feature capabilities of each Cisco UCS domain at the time of domain registration. This ability enables you to seamlessly integrate multiple versions of Cisco UCS Manager with Cisco UCS Central for management and global service profile deployment.

The Cisco UCS Central XML API is a programmatic interface to Cisco UCS Central. The API accepts XML documents through HTTP or HTTPS. Developers can use any programming language to generate XML documents that contain the API methods. Configuration and state information for Cisco UCS Central is stored in a hierarchical tree structure known as the management information tree, which is completely accessible through the XML API.

The Cisco UCS Central XML API supports operations on a single object or an object hierarchy. A single API call can make changes to a single attribute of an object such as the power state of a blade, or many objects such as chassis, blades, adapters, policies, and other configurable components.

The API operates in forgiving mode. Missing attributes are substituted with default values (if applicable) that are maintained in the internal data management engine (DME). The DME ignores incorrect attributes. If multiple managed objects (MOs) are being configured (for example, virtual NICs), and any of the MOs cannot be configured, the API stops its operation. It rolls back the management information tree to its prior state (before the operation) and returns an error.

Updates to MOs and properties conform to the existing object model, ensuring backward compatibility. If existing properties are changed during a product upgrade, they are managed during the database load after the upgrade. New properties are assigned default values.

Operation of the API is transactional and terminates on a single data model. Cisco UCS is responsible for all endpoint communication, such as state updates; users cannot communicate directly to endpoints. In this way, developers are relieved from the task of administering isolated, individual component configurations.

The API model includes the following programmatic entities:

- Classes—Define the properties and states of objects in the management information tree.
- Methods—Actions that the API performs on one or more objects.
- Types—Object properties that map values to the object state (for example, `equipmentPresence`).

A typical request comes into the DME and is placed in the transactor queue in FIFO order. The transactor gets the request from the queue, interprets the request, and performs an authorization check. After the request

is confirmed, the transactor updates the management information tree. This complete operation is done in a single transaction.

Full event subscription is enabled. After subscribing, any event notification is sent along with its type of state change.

Management Information Model

All the physical and logical components that comprise Cisco UCS are represented in a hierarchical management information model, referred to as the MIT. Each node in the tree represents a managed object (MO) or group of objects that contains its administrative state and its operational state.

The hierarchical structure starts at the topRoot (`topRoot`) and contains parent and child nodes. Each node in this tree is a managed object and each object in Cisco UCS has a unique distinguished name (DN) that describes the object and its place in the tree. Managed objects are abstractions of the Cisco UCS Central resources, such as global service profiles, global policies, and global MAC pools.

Configuration policies are the majority of the policies in the system and describe the configurations of different Cisco UCS components. Policies determine how the system behaves under specific circumstances. Certain managed objects are not created by users, but are automatically created by the Cisco UCS, for example, power supply objects and fan objects. By invoking the API, you are reading and writing objects to the management information model (MIM).

Illustration of MIM Structure Showing Five Chassis

```
Tree (topRoot):————————— Distinguished Name:
|———DomainContainer———— (compute)
  |———ucsDomain-1————— (compute/sys-1008/)
    |———chassis-1————— (compute/sys-1008/chassis-1)
      |———blade-1————— (compute/sys-1008/chassis-1/blade-1)
        |———adaptor-1——— (compute/sys-1008/chassis-1/blade-1/adaptor-1)
      |———blade-2————— (compute/sys-1008/chassis-1/blade-2)
        |———adaptor-1——— (compute/sys-1008/chassis-1/blade-2/adaptor-1)
        |———adaptor-2——— (compute/sys-1008/chassis-1/blade-2/adaptor-2)
      |———blade-3————— (compute/sys-1008/chassis-1/blade-3)
        |———adaptor-1——— (compute/sys-1008/chassis-1/blade-3/adaptor-1)
        |———adaptor-2——— (compute/sys-1008/chassis-1/blade-3/adaptor-2)
      |———blade-4————— (compute/sys-1008/chassis-1/blade-4)
        |———adaptor-1——— (compute/sys-1008/chassis-1/blade-4/adaptor-1)
      |———blade-5————— (compute/sys-1008/chassis-1/blade-5)
        |———adaptor-1——— (compute/sys-1008/chassis-1/blade-5/adaptor-1)
        |———adaptor-2——— (compute/sys-1008/chassis-1/blade-5/adaptor-2)
      |———blade-6————— (compute/sys-1008/chassis-1/blade-6)
        |———adaptor-1——— (compute/sys-1008/chassis-1/blade-6/adaptor-1)
      |———blade-7————— (compute/sys-1008/chassis-1/blade-7)
        |———adaptor-1——— (compute/sys-1008/chassis-1/blade-7/adaptor-1)
      |———blade-8————— (compute/sys-1008/chassis-1/blade-8)
        |———adaptor-1——— (compute/sys-1008/chassis-1/blade-8/adaptor-1)
    |———chassis-2————— (compute/sys-1008/chassis-2)
    |———chassis-3————— (compute/sys-1008/chassis-3)
    |———chassis-4————— (compute/sys-1008/chassis-4)
    |———chassis-5————— (compute/sys-1008/chassis-5)
  |———ucsDomain-2————— (compute/sys-1009/)
```

Central Manager

Currently, in Cisco UCS Central the client application needs to be aware of all the individual service provider DMEs such as resource manager or identifier manager, and know that to create a service profile, the request must be sent to the resource manager, and to create a pool, the request must be sent to the policy manager. Cisco UCS Central has over 1000 managed objects, therefore it is not possible for the clients to know which provider DME maps to which MO.

Effective with this release, The Central Manager provides a virtual DME as a common interface to the back end process. It can be viewed as a DME that receives requests and sends these to the actual DME, receives the results, and aggregates and returns the appropriate response. The Central Manager is implemented as a new DME that receives all the XML API requests from external users. XML API requests from the GUI and CLI clients however, are processed by the existing DMEs. The Central Manager receives the API requests to this URL location: http://ucs central IP/xmlIM

Cisco UCS Central contains the following service provider DMEs:

- Service Registry - provides a centralized registration repository, storing the system information of the service providers (Identifier Manager, Operation Manager and so on) and the service consumers or clients (UCS Managers).
- Operations Manager - provides operations management (managing the firmware packs, exporting or importing configuration, backing up the database) functionality for a set of UCS systems.
- Identifier Manager - centralized management for UUIDs, MAC addresses, WWxNs, IP addresses and IQN addresses across UCS Manager systems to avoid conflicts.
- Resource Manager - provides a centralized and consolidated view of physical and logical resources (such as service profiles, VLANs or VSANs) across UCS systems.
- Management Controller - controller for Cisco UCS Central virtual machine.
- Policy Manager - provides the global policies and global pools. Because these objects exist under 'org', the organization structure is also owned and managed by the policy server. ID pools, templates, and Domain Groups are also defined in Policy Manager and they are selectively distributed to different services based on used cases.
- Statistics Manager - collects and stores statistics from the registered UCS domains.

Cisco UCS Central XML API Sample Flow

A typical request comes into the data management engine (DME) and is placed in the transactor queue in FIFO order. The transactor gets the request from the queue, interprets the request, and performs an authorization check. After the request is confirmed, the transactor updates the management information tree. This operation is done in a single transaction.

Object Naming

You can identify a specific object by its distinguished name (DN) or by its relative name (RN).

Distinguished Name

The distinguished name enables you to unambiguously identify a target object. The distinguished name has the following format consisting of a series of relative names:

```
dn = {rn}/{rn}/{rn}/{rn}...
```

In the following example, the DN provides a fully qualified path for `adaptor-1` from the top of the object tree to the object. The DN specifies the exact managed object on which the API call is operating.

```
< dn =”sys/chassis-5/blade-2/adaptor-1” />
```

Relative Name

The relative name identifies an object within the context of its parent object. The distinguished name is composed of a sequence of relative names.

For example, this distinguished name:

```
<dn = "sys/chassis-5/blade-2/adaptor-1/host-eth-2"/>
```

is composed of the following relative names:

```
topSystem MO: rn="sys"
equipmentChassis MO: rn="chassis-<id>"
computeBlade MO: rn ="blade-<slotId>"
adaptorUnit MO: rn="adaptor-<id>"
adaptorHostEthIf MO: rn="host-eth-<id>"
```

API Method Categories

Each method corresponds to an XML document.

Note Several code examples in this guide substitute the term `<real_cookie>` for an actual cookie (such as 1217377205/85f7ff49-e4ec-42fc-9437-da77a1a2c4bf). The XML API cookie is a 47-character string; it is not the type of cookie that web browsers store locally to maintain session information.

Authentication Methods

Authentication methods authenticate and maintain the session. For example:

- aaaLogin—Initial method for logging in.
- aaaRefresh—Refreshes the current authentication cookie.
- aaaLogout—Exits the current session and deactivates the corresponding authentication cookie.

Use the aaaLogin method to get a valid cookie. Use aaaRefresh to maintain the session and keep the cookie active. Use the aaaLogout method to terminate the session (also invalidates the cookie). A maximum of 256 sessions to the Cisco UCS can be opened at any one time.

Query Methods

Query methods obtain information on the current configuration state of an object. The following are query methods supported in this release:

- configResolveDn—Retrieves objects by DN.
- configResolveDns—Retrieves objects by a set of DNs.
- configResolveClass—Retrieves objects of a given class.
- configResolveClasses—Retrieves objects of multiple classes.
- configFindDnsByClassId—Retrieves the DNs of a specified class.
- configResolveChildren—Retrieves the child objects of an object.
- configResolveParent—Retrieves the parent object of an object.
- configScope—Performs class queries on a DN in the management information tree.

Most query methods have the argument inHierarchical (Boolean true/yes or false/no). If true, the inHierarchical argument returns all child objects.

```
<configResolveDn ... inHierarchical="false"></>
<configResolveDn ... inHierarchical="true"></>
```

Because the amount of data returned from Cisco UCS can be quite large, the inHierarchical argument should be used with care. For example, if the query method is used on a class or DN that refers to a managed object (MO) that is located high on the management information tree and inHierarchical is set to true, the response can contain almost the entire Cisco UCS configuration. The resources required for Cisco UCS to process the request can be high, causing Cisco UCS to take an extended amount of time to respond. To avoid delays, the query method should be performed on a smaller scale involving fewer MOs.

Tip If a query method does not respond or is taking a long time to respond, increase the timeout period on the client application or adjust the query method to involve fewer MOs.

The query API methods might also have an inRecursive argument to specify whether the call should be recursive (for example, follow objects that point back to other objects or the parent object).

The API also provides a set of filters to increase the usefulness of the query methods. These filters can be passed as part of a query and are used to identify the wanted result set.

Note Until a host is powered on at least once, Cisco UCS may not have complete inventory and status information. For example, if Cisco UCS is reset, it will not have detailed CPU, memory, or adapter inventory information until the next time the host is powered on. If a query method is performed on a MO corresponding to the unavailable data, the response will be blank.

Simple Filters

There are two simple filters, the true filter and false filter. These two filters react to the simple states of true or false, respectively.

- True filter—Result set of objects with the Boolean condition of true.
- False filter—Result set of objects with the Boolean condition of false.

Property Filters

The property filters use the values of an object's properties as the criteria for inclusion in a result set. To create most property filters, `classId` and `propertyId` of the target object/property is required, along with a value for comparison.

- Equality filter—Restricts the result set to objects with the identified property of “equal” to the provided property value.
- Not equal filter—Restricts the result set to objects with the identified property of “not equal” to the provided property value.
- Greater than filter—Restricts the result set to objects with the identified property of “greater than” the provided property value.
- Greater than or equal filter—Restricts the result set to objects with the identified property of “is greater than or equal” to the provided property value.
- Less than filter—Restricts the result set to objects with the identified property of “less than” the provided property value.
- Less than or equal filter—Restricts the result set to objects with the identified property of “less than or equal” to the provided property value.
- Wildcard filter—Restricts the result set to objects with the identified property matches that includes a wildcard. Supported wildcards include “%” or “*” (any sequence of characters), “?” or “-” (any single character).
- Any bits filter—Restricts the result set to objects with the identified property that has at least one of the passed bits set. (Use only on bitmask properties.)
- All bits filter—Restricts the result set to objects with the identified property that has all the passed bits set. (Use only on bitmask properties.)

Composite Filters

The composite filters are composed of two or more component filters. They enable greater flexibility in creating result sets. For example, a composite filter could restrict the result set to only those objects that were accepted by at least one of the contained filters.

- AND filter—Result set must pass the filtering criteria of each component filter. For example, to obtain all compute blades with `totalMemory` greater than 64 megabytes and operability of operable, the filter is composed of one greater than filter and one equality filter.

- OR filter—Result set must pass the filtering criteria of at least one of the component filters. For example, to obtain all the service profiles that have an `assignmentState` of unassigned or an association state value of unassociated, the filter is composed of two equality filters.
- Between filter—Result set is those objects that fall between the range of the first specified value and second specified value, inclusive. For example, all faults that occurred starting on the first date and ending on the last date.
- XOR filter—Result set is those objects that pass the filtering criteria of no more than one of the composite's component filters.

Modifier Filter

A modifier filter changes the results of a contained filter.

The only modifier filter that is currently supported is the NOT filter that negates the result of a contained filter. Use this filter to obtain objects that do not match contained criteria.

Configuration Methods

There are several methods to make configuration changes to managed objects. These changes can be applied to the whole tree, a subtree, or an individual object. The following are examples of configuration methods:

- configConfMo—Affects a single managed object (for example, a DN) in the management information tree.
- configConfMos—Affects multiple subtrees (for example, several DNs).
- configConfMoGroup—Makes the same configuration changes to multiple subtree structures (DNs) or managed objects.

Most configuration methods use the argument inHierarchical (Boolean true/yes or false/no). These values do not play a significant role during configuration because child objects are included in the XML document and the DME operates in the forgiving mode.

Event Subscription Methods

Applications get state change information by regular polling or event subscription. For more efficient use of resources, event subscription is the preferred method of notification. Polling should be used only under very limited circumstances.

Use eventSubscribe to register for events, as shown the following example:

```
<eventSubscribe
    cookie="<real_cookie>">
</eventSubscribe>
```

To receive notifications, open an HTTP or HTTPS session over TCP and keep the session open. On receiving eventSubscribe, starts sending all new events as they occur. You can unsubscribe from these events using the eventUnsubscribe method.

Each event includes an srcDme attribute that indicates the application where that event is generated, and each event has a unique event ID for a given srcDme. Here, the srcDme attribute represents the source application

of the event. Event IDs operate as counters and are included in all method responses. When an event is generated, the event ID counter increments and is assigned as the new event ID. This sequential numbering enables tracking of events and ensures that no event is missed.

An event channel connection opened by a user will be closed automatically by after 600 seconds of inactivity associated with the event channel session cookie. To prevent automatic closing of the event channel connection by , the user must either send the aaaKeepAlive request for the same event channel session cookie within 600 seconds or send any other XML API method to using the same event channel session cookie.

Capturing XML Interchange Between the GUI and Cisco UCS Central

To capture the XML interchange between the GUI and Cisco UCS Central, use the Google Chrome web browser's developer tool shortcut Ctrl + Shift + I. Due to internal security requirements, this information is not always complete. However, you can use a commercial packet analyzer application to observe sent XML.

Success or Failure Response

When responds to an XML API request, the response indicates failure if the request is impossible to complete. A successful response indicates only that the request is valid, not that the operation is complete. For example, it may take some time for a server to finish a power-on request. The power state changes from down to up only after the server actually powers on.

Successful Requests

When a request has executed successfully, returns an XML document with the information requested or a confirmation that the changes were made. The following is an example of a configResolveDn query on the distinguished name `sys/chassis-1/blade-1`:

```
<configResolveDn
     dn="sys/chassis-1/blade-1"
     cookie="<real_cookie>"
     inHierarchical="false"/>
```

Failed Requests

The response to a failed request includes XML attributes for errorCode and errorDescr. The following is an example of a response to a failed request:

```
<configConfMo dn="fabric/server"
     cookie="<real_cookie>"
     response="yes"
     errorCode="103"
     invocationResult="unidentified-fail"
     errorDescr="can't create; object already exists.">
</configConfMo>
```

Empty Results

A query request for a nonexistent object is not treated as a failure by the DME. If the object does not exist, returns a success message, but the XML document contains an empty data field (`<outConfig> </outConfig>`)

to indicate that the requested object was not found. The following example shows the response to an attempt to resolve the distinguished name on a nonexistent rack-mount server:

```
<configResolveDn
     dn="sys/chassis-1/blade-4711"
     cookie="<real_cookie>"
     response="yes">
     <outConfig>
     </outConfig>
</configResolveDn>
```

CHAPTER 2

Cisco UCS Central XML API Methods

This chapter includes the following sections:

Authentication Methods

Authentication allows XML API interaction with . It provides a way to set permissions and control the operations that can be performed.

Note Most code examples in this guide substitute the term `<real_cookie>` for an actual cookie (such as 1217377205/85f7ff49-e4ec-42fc-9437-da77a1a2c4bf). The Cisco UCS cookie is a 47-character string; it is not the type of cookie that web browsers store locally to maintain session information.

Login

To log in, the XML API client establishes a TCP connection to the Cisco UCS Central HTTPS server and posts an XML document containing the aaaLogin method to the following URL:
`https://<UCSCENTRALIP>/xmlIM/`

Next, the client specifies the aaaLogin method and provides a user name and password:

```
<aaaLogin
    inName="admin"
    inPassword="password" />
```

Note Do not include XML version or DOCTYPE lines in the XML API document. The inName and inPassword attributes are parameters.

Each XML API document represents an operation to be performed. When the request is received as an XML API document, Cisco UCS reads the request and performs the actions as provided in the method. Cisco UCS responds with a message in XML document format and indicates success or failure of the request.

The following is a typical successful response:

```
1 <aaaLogin
2    response="yes"
3    outCookie="<real_cookie>"
4       outRefreshPeriod="600"
5       outPriv="aaa,ext-lan-policy,ext-lan-qos,ext-san-policy,operations,
                 pod-policy,pod-qos,read-only"
6    outDomains="mgmt02-dummy"
7    outChannel="noencssl"
8    outEvtChannel="noencssl">
9 </aaaLogin>
```

Each line in the response should be interpreted as follows:

1. Specifies the method used to login.
2. Confirms that this is a response.
3. Provides the session cookie.
4. Specifies the recommended cookie refresh period. The default login session length is 600 seconds.
5. Specifies the privilege level assigned to the user account.
6. The outDomains value is `mgmt02-dummy`.
7. The outChannel value of `noencssl` declares that this session is not using encryption over SSL.
8. The outEvtChannel value of `noencssl` declares that any event subscriptions would not use encryption over SSL.
9. Closing tag.

Alternatively, you can use the cURL utility to log in to the XML API. You must use HTTPS in the cURL command, as shown in the following example:

```
curl -d "<aaaLogin inName='admin' inPassword='password'></aaaLogin>"
https://192.0.20.72/xmlIM/
```

Refreshing the Session

Sessions are refreshed with the aaaRefresh method, using the 47-character cookie obtained either from the aaaLogin response or a previous refresh.

```
<aaaRefresh
     inName="admin"
     inPassword="mypasword"
     inCookie="real_cookie"/>
```

Logging Out of the Session

Use the following method to log out of a session:

```
<aaaLogout
     inCookie="<real_cookie>" />
```

Unsuccessful Responses

Failed login:

```
<aaaLogin
    response="yes"
    cookie="<real_cookie>"
    errorCode="551"
    invocationResult="unidentified-fail"
    errorDescr="Authentication failed">
</aaaLogin>
```

Nonexistent object (blank return indicates no object with the specified DN):

```
<configResolveDn
    dn="sys-machine/chassis-1/blade-4711“
    cookie="<real_cookie>“
    response="yes">
    <outConfig> </outConfig>
</configResolveDn>
```

Bad request:

```
<configConfMo
    dn="fabric/server“
    cookie="<real_cookie>“
    response="yes“
    errorCode="103“
    invocationResult="unidentified-fail“
    errorDescr="can't create; object already exists.">
</configConfMo>
```

Query Methods

Using configResolveChildren

When resolving children of objects in the MIT, note the following:

- This method obtains all child objects of a named object that are instances of the named class. If a class name is omitted, all child objects of the named object are returned.
- inDn attribute specifies the named object from which the child objects are retrieved (required).
- classId attribute specifies the name of the child object class to return (optional).
- Authentication cookie (from aaaLogin or aaaRefresh) is required.
- inHierarchical attribute (default = false) if true, specifies that results are hierarchical.
- Enumerated values, `classIds`, and bit masks are displayed as strings.

See the example request/response in configResolveChildren, on page 54.

Using configResolveClass

When resolving a class, note the following:

- All objects of the specified class type are retrieved.

- classId specifies the object class name to return (required).
- Authentication cookie (from aaaLogin or aaaRefresh) is required.
- inHierarchical attribute (default = false) if true, specifies that results are hierarchical.
- Enumerated values, `classIds`, and bit masks are displayed as strings.

Result sets can be large. Be precise when defining result sets. For example, to obtain only a list of servers, use computeItem as the attribute value for classId in the query. To get all instances of equipment, query the `equipmentItem` class. This example queries for all instances of the `equipmentItem` class:

```
<configResolveClass
 cookie="real_cookie"
 inHierarchical="false"
 classId="equipmentItem"/>
```

See the example request/response in configResolveClass, on page 56.

Using configResolveClasses

When resolving multiple classes, note the following:

- This method retrieves all the objects of the specified class types.
- classId attribute specifies the name of the object class to return (required).
- Authentication cookie (from aaaLogin or aaaRefresh) is required.
- inHierarchical attribute (default = false) if true, specifies that results are hierarchical.
- Enumerated values, `classIds`, and bit masks are displayed as strings.

If an invalid class name is specified in the inId attribute, an XML parsing error is generated and the query cannot execute.

See the example request/response in configResolveClasses, on page 61.

Using configResolveDn

When resolving a DN, note the following:

- The object specified by the DN is retrieved.
- Specified DN identifies the object instance to be resolved (required).
- Authentication cookie (from aaaLogin or aaaRefresh) is required.
- inHierarchical attribute (default = false) if true, specifies that results are hierarchical.
- Enumerated values, `classIds`, and bit masks are displayed as strings.

See the example request/response in configResolveDn, on page 63.

Using configResolveDns

When resolving multiple DNs, note the following:

- The objects specified by the DNs are retrieved.
- Specified DN identifies the object instance to be resolved (required).
- Authentication cookie (from aaaLogin or aaaRefresh) is required.
- inHierarchical attribute (default = false) if true, specifies that results are hierarchical.
- Enumerated values, `classIds`, and bit masks are displayed as strings.
- Order of a request does not determine the order of the response.
- Unknown DNs are returned as part of the `outUnresolved` element.

See the example request/response in configResolveDns, on page 64.

Using configResolveParent

When resolving the parent object of an object, note the following:

- This method retrieves the parent object of a specified DN.
- dn attribute is the DN of the child object (required).
- Authentication cookie (from aaaLogin or aaaRefresh) is required.
- inHierarchical attribute (default = false) if true, specifies that results are hierarchical.
- Enumerated values, `classIds`, and bit masks are displayed as strings.

See the example request/response in configResolveParent, on page 66.

Using configScope

Limiting the scope of a query allows for a finer grained, less resource-intensive request. The query can be anchored at a point in the management information tree other than the root. When setting the query scope, note the following:

- This method sets the root (scope) of the query to a specified DN and returns objects of the specified class type.
- dn is the named object from which the query is scoped (required).
- inClass attribute specifies the name of the object class to return (optional; when a class is not specified, the query acts the same as configResolveDn).
- Authentication cookie (from aaaLogin or aaaRefresh) is required.
- inHierarchical attribute (default = false) if true, specifies that results are hierarchical.
- Enumerated values, `classIds`, and bit masks are displayed as strings.

The following example is a query for the Ethernet interfaces on the blades in chassis 1:

```
<configScope
    dn="sys/chassis-1"
    inClass="adaptorExtEthIf"
    cookie="<real_cookie>"
    inHierarchical="false"/>
```

Also see the example request/response in configScope, on page 67.

Querying the Mac Pool

To obtain a list of all MAC addresses, query for macpoolAddr. These are children of the (system-created) `macpoolUniverse`. The request is as follows:

```
<configScope
    cookie="<real_cookie>"
    inHierarchical="false"
    dn="mac" inClass="macpoolAddr"/>
```

The response is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<configScope
    cookie="<real_cookie>"
    dn="mac" response="yes">
    <outConfigs>
       <macpoolAddr
          assigned="no"
          assignedToDn=""
          dn="mac/00:00:00:00:FF:0F"
          id="00:00:00:00:FF:0F"
          owner="pool">
       </macpoolAddr>
       <macpoolAddr
          assigned="no"
          assignedToDn=""
          dn="mac/00:00:00:00:FF:0E"
          id="00:00:00:00:FF:0E"
          owner="pool">
       </macpoolAddr>
       .
       .
       .
    <\outconfig
<\configScope
```

Because the objects of the `macpoolAddr` class can exist only as children of the MAC pool universe, a simpler query follows:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="macpoolAddr"/>
```

To determine which `computeItem` (blade or rack mount server) is assigned a particular MAC address, specify the MAC address in the query and look at the `assignedToDn` field in the response. For example, a request with a specified MAC address follows:

```
<configResolveDn
    cookie="<real_cookie>"
    inHierarchical="false"
```

```
    dn="mac/10:00:00:00:00:03"/>
```

The response is as follows:

```
<configResolveDn
    dn="mac/10:00:00:00:00:03"
    cookie="real_cookie"
    response="yes">
    <outConfig>
       <macpoolAddr assigned="yes"
          assignedToDn="org-root/ls-BOB/ether-eth1"
          dn="mac/10:00:00:00:00:03" id="10:00:00:00:00:03"
          owner="pool" />
    </outConfig>
</configResolveDn>
```

Querying Faults

The following example obtains a list of major faults:

```
<configResolveClass
    cookie="real_cookie"
    inHierarchical="false"
    classId="faultInst"/>
```

The following example (which contains the filter `<inFilter>eq class= “faultInst”`) obtains a list of all major or critical faults:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="faultInst">
    <inFilter>
       <eq class="faultInst"
          property="highestSeverity"
          value="major" />
    </inFilter>
</configResolveClass>
```

Using Filters

Simple Filters

Simple filters use true and false conditions of properties to select results.

False example:

```
<configResolveClass
    cookie="<real_cookie>"
    classId="topSystem"
    inHierarchical="false">
    <inFilter>
    </inFilter>
</configResolveClass>
```

True example:

```
<configResolveClass
    cookie="<real_cookie>"
    classId="topSystem"
    inHierarchical="true">
    <inFilter>
    </inFilter>
</configResolveClass>
```

Property Filters

Equality Filter

The example shows a query for all associated servers. The filter is framed as follows:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="lsServer">
    <inFilter>
       <eq class="lsServer"
          property="assocState"
          value="associated" />
    </inFilter>
</configResolveClass>
```

Not Equal Filter

The example finds all unassigned servers (assignment state property is not equal to assigned). The filter is framed as follows:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="lsServer">
    <inFilter>
       <ne class="lsServer"
          property="assignState"
          value="assigned" />
    </inFilter>
</configResolveClass>
```

Greater Than Filter

The example finds the memory arrays with more than 1024 MB capacity. The filter is framed as follows:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="memoryArray">
    <inFilter>
       <gt class="memoryArray"
          property="currCapacity"
          value="1024" />
    </inFilter>
```

```
</configResolveClass>
```

Greater Than or Equal to Filter

The example finds the memory arrays with 2048 MB capacity or more. The filter is framed as follows:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="memoryArray">
    <inFilter>
       <ge class="memoryArray"
          property="currCapacity"
          value="2048" />
    </inFilter>
</configResolveClass>
```

Less Than Filter

The example finds memory arrays with less than 1024 MB capacity. The filter is framed as follows:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="memoryArray">
    <inFilter>
       <lt class="memoryArray"
          property="currCapacity"
          value="1024" />
    </inFilter>
</configResolveClass>
```

Less Than or Equal to Filter

The example finds memory arrays with 2048 MB capacity or less. The filter is framed as follows:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="memoryArray">
    <inFilter>
       <le class="memoryArray"
          property="currCapacity"
          value="2048" />
    </inFilter>
</configResolveClass>
```

Wildcard Filter

The wildcard filter uses standard regular expression syntax. The example finds any adapter unit whose serial number begins with the prefix QCI1:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="adaptorUnit">
```

```
    <inFilter>
       <wcard class="adaptorUnit"
          property="serial"
          value="QCI1.*" />
    </inFilter>
</configResolveClass>
```

Any Bits Filter

This example finds all servers that have a `connStatus` of either A or B (the property `connStatus` is a bit mask). The filter is framed as follows:

```
<configResolveClass
    cookie="null"
    inHierarchical="false"
    classId="computeItem">
    <inFilter>
       <anybit class="computeItem"
          property="connStatus"
          value="A,B" />
    </inFilter>
</configResolveClass>
```

All Bits Filter

The example finds all service profiles with the `configQualifier` bit mask set to both vnic-capacity and vhba-capacity. The filter is framed as follows:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="lsServer">
    <inFilter>
       <allbits class="lsServer"
          property="configQualifier"
          value="vnic-capacity,vhba-capacity" />
    </inFilter>
</configResolveClass>
```

Composite Filters

AND Filter

To determine all UUIDs assigned and owned by pools, run the following query. The filter is framed as follows:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="uuidpoolAddr">
    <inFilter>
       <and>
          <eq class="uuidpoolAddr"
             property="owner"
             value="pool" />
          <eq class="uuidpoolAddr"
             property="assigned"
             value="yes" />
```

```
      </and>
    </inFilter>
</configResolveClass>
```

The response is as follows:

```
<configResolveClass
    classId="uuidpoolAddr"
    cookie="<real_cookie>"
    response="yes">
    <outConfigs>
       <uuidpoolAddr
          assigned="yes"
          assignedToDn="org-root/ls-foo"
          dn="uuid/F000-00000000000F"
          id="F000-00000000000F"
          owner="pool">
       </uuidpoolAddr>
    </outConfigs>
</configResolveClass>
```

In the example, the AND filter finds the chassis with vendor Cisco Systems Inc and serial number CHS A04:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="equipmentChassis">
    <inFilter>
       <and>
          <eq property="vendor"
             value="Cisco Systems Inc"
             class="equipmentChassis"/>
          <eq class="equipmentChassis"
             property="serial"
             value="CHS A04"/>
       </and>
    </inFilter>
</configResolveClass>
```

OR Filter

The example returns all objects of type `computeItem` that are located in slot one or slot eight from all chassis.

```
<configResolveClass
    inHierarchical="false"
    cookie="<real_cookie>"
    classId="compute">
    <inFilter>
       <or>
          <eq class="computeItem"
             property="slotId"
             value="1"/>
          <eq class="computeItem"
             property="slotId"
             value="8"/>
        </or>
    </inFilter>
</configResolveClass>
```

Between Filter

The example finds the memory arrays with slots 1, 2, 3, 4, or 5 populated (note that the between range is inclusive). The filter is framed as follows:

```
<configResolveClass
    cookie="<real_cookie>"
    inHierarchical="false"
    classId="memoryarray">
    <inFilter>
       <bw class="memoryArray"
          property="populated"
          firstValue="1"
          secondValue="5"/>
    </inFilter>
</configResolveClass>
```

AND, OR, NOT Composite Filter

The example is an AND, OR, NOT combination. It returns all objects of the `computeItem` type that are located in slot one or slot eight from all chassis, except chassis five.

```
<configResolveClass
    inHierarchical="false"
    cookie="<real_cookie>"
    classId="computeItem">
    <inFilter>
       <and>
          <or>
             <eq class="computeItem" property="slotId" value="1"/>
             <eq class="computeItem" property="slotId" value="8"/>
          </or>
          <not>
             <eq class="computeItem" property="chassisId" value="5"/>
          </not>
       </and>
    </inFilter>
</configResolveClass>
```

NOT Modifier Filter

The NOT filter can negate a contained filter. The filter is framed as follows. The example queries for servers that do not have a `connStatus` of unknown (the property `connStatus` is a bit mask).

```
<configResolveClass
    cookie="null"
    inHierarchical="false"
    classId="computeItem">
    <inFilter>
       <not>
          <anybit class="computeItem"
             property="connStatus"
             value="unknown" />
       </not>
    </inFilter>
</configResolveClass>
```

CHAPTER 3

Cisco UCS Central XML API Method Descriptions

This chapter includes the following sections:

aaaChangeSelfPassword

The aaaChangeSelfPassword method changes the user's own password. The user supplies the old password for authentication, the new password, and a confirmation of the new password. If the user is authenticated successfully with the old password, the new password becomes effective.

Note Users with admin or aaa privilege are not required to provide the old password while using this method.

Request Syntax

```
<xs:element name="aaaChangeSelfPassword" type="aaaChangeSelfPassword"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaChangeSelfPassword" mixed="true">
            <xs:attribute name="inUserName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inOldPassword">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:minLength value="0"/>
                         <xs:maxLength value="510"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inNewPassword">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:minLength value="0"/>
                         <xs:maxLength value="510"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inConfirmNewPassword">
                <xs:simpleType>
```

```
                    <xs:restriction base="xs:string">
                         <xs:minLength value="0"/>
                         <xs:maxLength value="510"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaChangeSelfPassword" type="aaaChangeSelfPassword"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaChangeSelfPassword" mixed="true">
            <xs:attribute name="outStatus">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="success"/>
                         <xs:enumeration value="failure"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Changing Your Password

Request

```
<aaaChangeSelfPassword
    cookie="<real_cookie>"
    inUserName="admin"
    inOldPassword="Nbv12345"
    inNewPassword="Mbv12345"
    inConfirmNewPassword="Mbv12345" />
```

Response

```
<aaaChangeSelfPassword
    cookie="<real_cookie>"
    response="yes"
    outStatus="success">
</aaaChangeSelfPassword>
```

aaaCheckComputeAuthToken

The aaaCheckComputeAuthToken method gets details on the specified token, such as the user name (who generated this token) and the user's privileges and locales.

Request Syntax

```
<xs:element name="aaaCheckComputeAuthToken" type="aaaCheckComputeAuthToken"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaCheckComputeAuthToken" mixed="true">
            <xs:attribute name="inUser">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inToken" type="xs:string"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaCheckComputeAuthToken" type="aaaCheckComputeAuthToken"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaCheckComputeAuthToken" mixed="true">
            <xs:attribute name="outAllow">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>

            <xs:attribute name="outRemote">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outAuthUser">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outLocales" type="xs:string"/>
            <xs:attribute name="outPriv">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern
value="((ext-lan-policy|pn-maintenance|ls-security-policy|pod-security|pn-equipment|ls-con
fig-policy|ext-san-policy|ls-security|aaa|power-mgmt|read-only|ext-lan-security|ls-config|
ls-server-policy|pod-qos|pn-policy|ls-storage-policy|admin|ext-san-security|pod-config|ls-
server|ext-lan-qos|ls-storage|ls-qos-policy|operations|ext-lan-config|pn-security|ls-netwo
rk-policy|pod-policy|ext-san-qos|ls-qos|ls-server-oper|ext-san-config|ls-network|ls-ext-ac
cess|fault),){0,35}(ext-lan-policy|pn-maintenance|ls-security-policy|pod-security|pn-equip
ment|ls-config-policy|ext-san-policy|ls-security|aaa|power-mgmt|read-only|ext-lan-security
|ls-config|ls-server-policy|pod-qos|pn-policy|ls-storage-policy|admin|ext-san-security|pod
-config|ls-server|ext-lan-qos|ls-storage|ls-qos-policy|operations|ext-lan-config|pn-securi
ty|ls-network-policy|pod-policy|ext-san-qos|ls-qos|ls-server-oper|ext-san-config|ls-networ
```

```
k|ls-ext-access|fault){0,1}"/>
                </xs:restriction>
            </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
 </xs:complexType>
```

Example: Verifying a User Token

Request

```
<aaaCheckComputeAuthToken
    cookie="<real_cookie>"
    inToken="0454187530930229968721 1"
    inUser="admin"/>
```

Response

```
<aaaCheckComputeAuthToken
    cookie="<real_cookie>"
    response="yes"
    outAllow="yes"
    outRemote="no"
    outAuthUser="admin"
    outLocales=""
    outPriv="admin,read-only">
</aaaCheckComputeAuthToken>
```

aaaCheckComputeExtAccess

The aaaCheckComputeExtAccess method validates whether a specified user has access to the server specified with the inDn parameter.

Request Syntax

```
<xs:element name="aaaCheckComputeExtAccess" type="aaaCheckComputeExtAccess"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaCheckComputeExtAccess" mixed="true">
            <xs:attribute name="inDn" type="referenceObject"/>
            <xs:attribute name="inUser">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaCheckComputeExtAccess" type="aaaCheckComputeExtAccess"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaCheckComputeExtAccess" mixed="true">
            <xs:attribute name="outAllow">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
 </xs:complexType>
```

Example: Validating User Access for a Specified Server

Request

```
<aaaCheckComputeExtAccess
    cookie="<real_cookie>"
    inDn="sys/Chassis-1/blade-2"
    inUser="gopis"/>
```

Response

```
<aaaCheckComputeExtAccess
    cookie="<real_cookie>"
    response="yes"
    outAllow="no">
</aaaCheckComputeExtAccess>
```

aaaGetNComputeAuthTokenByDn

The aaaGetNComputeAuthTokenByDn method returns the authentication tokens for TokenLogin to a particular server specified by DN.

Request Syntax

```
<xs:element name="aaaGetNComputeAuthTokenByDn" type="aaaGetNComputeAuthTokenByDn"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaGetNComputeAuthTokenByDn" mixed="true">
 <xs:attribute name="inDn">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:minLength value="0"/>
                         <xs:maxLength value="510"/>
                    </xs:restriction>
```

```
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inNumberOf" type="xs:unsignedByte"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaGetNComputeAuthTokenByDn" type="aaaGetNComputeAuthTokenByDn"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaGetNComputeAuthTokenByDn" mixed="true">
            <xs:attribute name="outUser">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outTokens" type="xs:string"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Requesting Token Information

Request

```
<aaaGetNComputeAuthTokenByDn
    cookie="<real_cookie>"
    inDn="sys/chassis-1/blade-2"
    inNumberOf="5"/>
```

Response

```
<aaaGetNComputeAuthTokenByDn
cookie="<real_cookie>"
response="yes"
outUser="__computeToken__"
outTokens="35505994195216127267211,93595551908527060232451,11769973096057301593991,527
2953867276549184403 1,7310664396999028091979 1">
</aaaGetNComputeAuthTokenByDn>
```

aaaKeepAlive

The aaaKeepAlive method keeps the session active until the default session time expires, using the same cookie after the method call.

Request Syntax

```
<xs:element name="aaaKeepAlive" type="aaaKeepAlive" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaKeepAlive" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
```

```
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaKeepAlive" type="aaaKeepAlive" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaKeepAlive" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Keeping a Session Active

Request

```
<aaaKeepAlive
    cookie="<real_cookie>" />
```

Response

```
<aaaKeepAlive
    cookie="<real_cookie>"
    commCookie="11/15/0/2969"
    srcExtSys="10.193.33.109"
    destExtSys="10.193.33.109"
    srcSvc="sam_extXMLApi"
    destSvc="mgmt-controller_dme"
    response="yes">
</aaaKeepAlive>
```

aaaLogin

The aaaLogin method is the login process and is required to begin a session. This action establishes the HTTP (or HTTPS) session between the client and Cisco UCS Central.

Request Syntax

```
<xs:element name="aaaLogin" type="aaaLogin" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaLogin" mixed="true">
            <xs:attribute name="inName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inPassword">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:minLength value="0"/>
                         <xs:maxLength value="510"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaLogin" type="aaaLogin" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaLogin" mixed="true">
            <xs:attribute name="outCookie" type="xs:string"/>
            <xs:attribute name="outRefreshPeriod" type="xs:unsignedInt"/>
            <xs:attribute name="outPriv">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern
value="((ext-lan-policy|pn-maintenance|ls-security-policy|pod-security|pn-equipment|ls-con
fig-policy|ext-san-policy|ls-security|aaa|power-mgmt|read-only|ext-lan-security|ls-config|
ls-server-policy|pod-qos|pn-policy|ls-storage-policy|admin|ext-san-security|pod-config|ls-
server|ext-lan-qos|ls-storage|ls-qos-policy|operations|ext-lan-config|pn-security|ls-netwo
rk-policy|pod-policy|ext-san-qos|ls-qos|ls-server-oper|ext-san-config|ls-network|ls-ext-ac
cess|fault),){0,35}(ext-lan-policy|pn-maintenance|ls-security-policy|pod-security|pn-equip
ment|ls-config-policy|ext-san-policy|ls-security|aaa|power-mgmt|read-only|ext-lan-security
|ls-config|ls-server-policy|pod-qos|pn-policy|ls-storage-policy|admin|ext-san-security|pod
-config|ls-server|ext-lan-qos|ls-storage|ls-qos-policy|operations|ext-lan-config|pn-securi
ty|ls-network-policy|pod-policy|ext-san-qos|ls-qos|ls-server-oper|ext-san-config|ls-networ
k|ls-ext-access|fault){0,1}"/>
                </xs:restriction>
            </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outDomains" type="xs:string"/>
            <xs:attribute name="outChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outEvtChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outSessionId">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:minLength value="0"/>
                         <xs:maxLength value="32"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outVersion" type="xs:string"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Logging In

Request

```
<aaaLogin
    inName="admin"
    inPassword="RU123B45"/>
```

Response

```
<aaaLogin
    cookie=""
    response="yes"
    outCookie="<real_cookie>"
    outRefreshPeriod="600"
    outPriv="admin,read-only"
    outDomains=""
    outChannel="noencssl"
    outEvtChannel="noencssl"
    outSessionId="web_41246_A"
    outVersion="1.4(0.61490)">
</aaaLogin>
```

aaaLogout

The aaaLogout method is a process to close a web session by passing the session cookie as input. It is not automatic. The user has to explicitly invoke the aaaLogout method to terminate the session.

Request Syntax

```
<xs:element name="aaaLogout" type="aaaLogout" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaLogout" mixed="true">
            <xs:attribute name="inCookie" type="xs:string"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaLogout" type="aaaLogout" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaLogout" mixed="true">
            <xs:attribute name="outStatus">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="success"/>
                         <xs:enumeration value="failure"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Logging Out

Request

```
<aaaLogout
    inCookie="<real_cookie>"/>
```

Response

```
<aaaLogout
    cookie=""
    response="yes"
```

```
    outStatus="success">
</aaaLogout>
```

aaaRefresh

The aaaRefresh method keeps sessions active (within the default session time frame) by user activity. There is a default of 600 seconds that counts down when inactivity begins. If the 600 seconds expire, Cisco UCS Central enters a sleep mode. It requires signing back in, which restarts the countdown. It continues using the same session ID.

Note Using this method expires the previous cookie and issues a new cookie.

Request Syntax

```
<xs:element name="aaaRefresh" type="aaaRefresh" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaRefresh" mixed="true">
            <xs:attribute name="inName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inPassword">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:minLength value="0"/>
                         <xs:maxLength value="510"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inCookie" type="xs:string"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaRefresh" type="aaaRefresh" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaRefresh" mixed="true">
            <xs:attribute name="outCookie" type="xs:string"/>
            <xs:attribute name="outRefreshPeriod" type="xs:unsignedInt"/>
            <xs:attribute name="outPriv">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern
value="((ext-lan-policy|pn-maintenance|ls-security-policy|pod-security|pn-equipment|ls-con
fig-policy|ext-san-policy|ls-security|aaa|power-mgmt|read-only|ext-lan-security|ls-config|
ls-server-policy|pod-qos|pn-policy|ls-storage-policy|admin|ext-san-security|pod-config|ls-
server|ext-lan-qos|ls-storage|ls-qos-policy|operations|ext-lan-config|pn-security|ls-netwo
rk-policy|pod-policy|ext-san-qos|ls-qos|ls-server-oper|ext-san-config|ls-network|ls-ext-ac
cess|fault),){0,35}(ext-lan-policy|pn-maintenance|ls-security-policy|pod-security|pn-equip
ment|ls-config-policy|ext-san-policy|ls-security|aaa|power-mgmt|read-only|ext-lan-security
|ls-config|ls-server-policy|pod-qos|pn-policy|ls-storage-policy|admin|ext-san-security|pod
-config|ls-server|ext-lan-qos|ls-storage|ls-qos-policy|operations|ext-lan-config|pn-securi
ty|ls-network-policy|pod-policy|ext-san-qos|ls-qos|ls-server-oper|ext-san-config|ls-networ
k|ls-ext-access|fault){0,1}"/>
                </xs:restriction>
            </xs:simpleType>
```

```
            </xs:attribute>
            <xs:attribute name="outDomains" type="xs:string"/>
            <xs:attribute name="outChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outEvtChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Refreshing the Session

Request

```
<aaaRefresh
    cookie="<real_cookie>"
    inName="admin"
    inPassword="RU123B45"
    inCookie="<real_cookie>"/>
```

Response

```
<aaaRefresh
    cookie="<real_cookie>"
    commCookie="" srcExtSys="0.0.0.0"
    destExtSys="0.0.0.0"
    srcSvc=""
    destSvc=""
    response="yes"
    outCookie="<real_cookie>"
    outRefreshPeriod="7200"
    outPriv="admin"
    outDomains=""
    outChannel="fullssl"
    outEvtChannel="fullssl">
</aaaRefresh>
```

aaaTokenLogin

The aaaTokenLogin method allows access to the user based on the token passed. These tokens authenticate the user instead of using the password to allow access to the system. Tokens are generated by aaaGetNComputeAuthToken method.

Request Syntax

```
<xs:element name="aaaTokenLogin" type="aaaTokenLogin" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaTokenLogin" mixed="true">
            <xs:attribute name="inName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inToken">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:minLength value="0"/>
                         <xs:maxLength value="510"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaTokenLogin" type="aaaTokenLogin" substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaTokenLogin" mixed="true">
            <xs:attribute name="outCookie" type="xs:string"/>
            <xs:attribute name="outRefreshPeriod" type="xs:unsignedInt"/>
            <xs:attribute name="outPriv">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern
value="((ext-lan-policy|pn-maintenance|ls-security-policy|pod-security|pn-equipment|ls-con
fig-policy|ext-san-policy|ls-security|aaa|power-mgmt|read-only|ext-lan-security|ls-config|
ls-server-policy|pod-qos|pn-policy|ls-storage-policy|admin|ext-san-security|pod-config|ls-
server|ext-lan-qos|ls-storage|ls-qos-policy|operations|ext-lan-config|pn-security|ls-netwo
rk-policy|pod-policy|ext-san-qos|ls-qos|ls-server-oper|ext-san-config|ls-network|ls-ext-ac
cess|fault),){0,35}(ext-lan-policy|pn-maintenance|ls-security-policy|pod-security|pn-equip
ment|ls-config-policy|ext-san-policy|ls-security|aaa|power-mgmt|read-only|ext-lan-security
|ls-config|ls-server-policy|pod-qos|pn-policy|ls-storage-policy|admin|ext-san-security|pod
-config|ls-server|ext-lan-qos|ls-storage|ls-qos-policy|operations|ext-lan-config|pn-securi
ty|ls-network-policy|pod-policy|ext-san-qos|ls-qos|ls-server-oper|ext-san-config|ls-networ
k|ls-ext-access|fault){0,1}"/>
                </xs:restriction>
            </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outDomains" type="xs:string"/>
            <xs:attribute name="outChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outEvtChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outSessionId">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
```

```
                        <xs:minLength value="0"/>
                        <xs:maxLength value="32"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outVersion" type="xs:string"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Logging in with a Token

Request

```
<aaaTokenLogin
    inName="admin"
    inToken="8027850296441080579135 1" />
```

Response

```
<aaaTokenLogin cookie=""
    response="yes"
    outCookie="<real_cookie>"
    outRefreshPeriod="600"
    outPriv="admin,read-only"
    outDomains=""
    outChannel="noencssl"
    outEvtChannel="noencssl"
    outSessionId="web_49374_A"
    outVersion="1.4(0.61490)">
</aaaTokenLogin>
```

aaaTokenRefresh

The aaaTokenRefresh method refreshes the current TokenLogin session.

Request Syntax

```
<xs:element name="aaaTokenRefresh" type="aaaTokenRefresh"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaTokenRefresh" mixed="true">
            <xs:attribute name="inName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inCookie" type="xs:string"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="aaaTokenRefresh" type="aaaTokenRefresh"
substitutionGroup="externalMethod"/>
        <xs:complexType name="aaaTokenRefresh" mixed="true">
            <xs:attribute name="outCookie" type="xs:string"/>
            <xs:attribute name="outRefreshPeriod" type="xs:unsignedInt"/>
            <xs:attribute name="outPriv">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern
value="((ext-lan-policy|pn-maintenance|ls-security-policy|pod-security|pn-equipment|ls-con
fig-policy|ext-san-policy|ls-security|aaa|power-mgmt|read-only|ext-lan-security|ls-config|
ls-server-policy|pod-qos|pn-policy|ls-storage-policy|admin|ext-san-security|pod-config|ls-
server|ext-lan-qos|ls-storage|ls-qos-policy|operations|ext-lan-config|pn-security|ls-netwo
rk-policy|pod-policy|ext-san-qos|ls-qos|ls-server-oper|ext-san-config|ls-network|ls-ext-ac
cess|fault),){0,35}(ext-lan-policy|pn-maintenance|ls-security-policy|pod-security|pn-equip
ment|ls-config-policy|ext-san-policy|ls-security|aaa|power-mgmt|read-only|ext-lan-security
|ls-config|ls-server-policy|pod-qos|pn-policy|ls-storage-policy|admin|ext-san-security|pod
-config|ls-server|ext-lan-qos|ls-storage|ls-qos-policy|operations|ext-lan-config|pn-securi
ty|ls-network-policy|pod-policy|ext-san-qos|ls-qos|ls-server-oper|ext-san-config|ls-networ
k|ls-ext-access|fault){0,1}"/>
                </xs:restriction>
            </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outDomains" type="xs:string"/>
            <xs:attribute name="outChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="outEvtChannel">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="fullssl"/>
                         <xs:enumeration value="noencssl"/>
                         <xs:enumeration value="plain"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Refreshing the Session with a Token

Request

```
<aaaTokenRefresh
    inName="admin"
    inCookie="<real_cookie>" />
```

Response

```
<aaaTokenRefresh
    cookie=""
```

```
    response="yes"
    outCookie="<real_cookie>"
    outRefreshPeriod="600"
    outPriv="admin,read-only"
    outDomains=""
    outChannel="noencssl"
    outEvtChannel="noencssl">
</aaaTokenRefresh>
```

configConfMo

The configConfMo method configures the specified managed object in a single subtree (for example, DN).

Request Syntax

```
<xs:element name="configConfMo" type="configConfMo" substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfMo" mixed="true">
            <xs:all>
                <xs:element name="inConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configConfMo" type="configConfMo" substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfMo" mixed="true">
            <xs:all>
                <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example: Configuring Specific Objects

Request

```
<configConfMo
    dn=""
    cookie="<real_cookie>"
```

```
    inHierarchical="false">
    <inConfig>
       <aaaLdapEp
          attribute="CiscoAvPair"
          basedn="dc=pasadena,dc=cisco,dc=com"
          descr=""
          dn="sys/ldap-ext"
          filter="sAMAccountName=$userid"
          retries="1"
          status="modified"
          timeout="30"/>
    </inConfig>
</configConfMo>
```

Response

```
<configConfMo
    dn=""
    cookie="<real_cookie>"
    commCookie="11/15/0/28"
    srcExtSys="10.193.33.101"
    destExtSys="10.193.33.101"
    srcSvc="sam_extXMLApi"
    destSvc="mgmt-controller_dme"
    response="yes">
    <outConfig>
       <aaaLdapEp
          attribute="CiscoAvPair"
          basedn="dc=pasadena,dc=cisco,dc=com"
          childAction="deleteNonPresent"
          descr=""
          dn="sys/ldap-ext"
          filter="sAMAccountName=$userid"
          fsmDescr=""
          fsmPrev="updateEpSuccess"
          fsmProgr="100"
          fsmStageDescr=""
          fsmStamp="2010-11-22T23:41:01.826"
          fsmStatus="nop"
          fsmTry="0"
          intId="10027"
          name=""
          retries="1"
          status="modified"
          timeout="30"/>
    </outConfig>
</configConfMo>
```

Example: Tagging Domains

When you schedule infrastructure firmware updates, you can schedule them for specific domains, or domains assigned to a domain group, using maintenance groups and tags.

You can apply a maintenance group tag to a domain with configConfMo. See the UCS Central Administration Guide for more information on Infrastructure Firmware updates.

Request

```
<configConfMo
    dn="holder/tag-ep"
    cookie="<real_cookie>"
    inHierarchical="false">
    <inConfig>
       <tagInstance
```

```
          defName="Maintenance Group"       // Type of tag
          taggedObjectDn="compute/sys-1009" // Domain name or ID
          value="tagTest"                   // Tag name
          status="created"
          />
    </inConfig>
</configConfMo>
```

Response

```
<configConfMo
  dn="holder/tag-ep"
  cookie="<real_cookie>"
  commCookie="18/16/0/a4a"
  userContext="no"
  srcExtSys="10.193.219.120"
  destExtSys="10.193.219.120"
  srcSvc="sam_extXMLApi"
  destSvc="central-mgr_dme"
  response="yes">
<outConfig>
<tagInstance
  defDn="holder/tag-def-ep/type-Maintenance Group"
  defName="Maintenance Group"
  dn="holder/tag-ep/type-Maintenance Group-inst-[tagTest]-obj-[compute/sys-1009]"
  objectName="StorMagicFI-A"
  objectType="computeSystem"
  srcDme="resource-mgr"
  status="created"
  taggedObjectDn="compute/sys-1009"
  value="tagTest"/>Config>
</configConfMo>
```

Creating a Hardware Compatibility List

Creating the Hardware Compatibility list involves multiple steps:

1. Creating an OS tag on the target server.
2. Creating an Adapter Driver tag on the target server.
3. Deleting the tags, when finished.

Example: Creating an OS tag on the Target Server

The Hardware Compatibility report queries the OS tagSoftwareInst tag.

To find the OS vendor and version, use configResolveClassIdx. For more information, see Example: Verifying the OS Vendor and Version, on page 59.

Request

```
<configConfMo
dn="holder/tag-ep"
cookie="<real_cookie>"
inHierarchical="false">
  <inConfig>
    <tagSoftwareInst
    defName="Operating System for HCR"      //required field
    taggedObjectDn="compute/sys1/ch1/bl3"   //required field; DN for the server
    value=""
    vendor="CentOS"                //required field; UCS Central OS
```

```
    version="CentOS 6.6"           //required field; UCS Central OS version
    status="created"
    />
    </inConfig>
</configConfMo>
```

Response

```
<configConfMo
dn="holder/tag-ep"
cookie="<real_cookie>"
commCookie="18/16/0/47b"
userContext="no"
srcExtSys="10.193.219.120"
destExtSys="10.193.219.120"
srcSvc="sam_extXMLApi"
destSvc="central-mgr_dme"
response="yes">
    <outConfig>
        <tagSoftwareInst
  defDn="holder/tag-def-ep/type-Operating System for HCR"
  defName="Operating System for HCR"
  dn="holder/tag-ep/type-Operating System for HCR-inst-[CentOS
6.6]-obj-[compute/sys1/ch1/bl3]"
  objectName=""
  objectType="computeBlade"
  srcDme="resource-mgr"
  status="created"
  taggedObjectDn="compute/sys1/ch1/bl3"
  value="CentOS 6.6"
  vendor="CentOS"
  version="CentOS 6.6"/>
    </outConfig>
</configConfMo>
```

Example: Creating an Adapter tag on the Target Server

The Hardware Compatibility report queries the Adapter Driver tagDriver tag.

Note If you need to verify different driver types, create multiple tagDrivers.

To find the adapter vendor and version, use configResolveClassIdx. For more information, see Example: Verifying the Adapter Vendor and Version, on page 60.

Request

```
<configConfMo
dn="holder/tag-ep"
cookie="<real_cookie>"
inHierarchical="false">
  <inConfig>
    <tagDriver
       defName='Adapter Driver for HCR' //required field
       taggedObjectDn='compute/sys1/ch1/bl4/ad1'  //required field
       protocol='Ethernet'
       vendor='Cisco' //required field; adapter vendor
       version='2.3.0.20' //required field; adapter driver version
       value=''
       status='created'
      />
    </inConfig>
</configConfMo>
```

Response

```
<configConfMo
dn="holder/tag-ep"
cookie="<real_cookie>"
commCookie="18/16/0/480"
userContext="no"
srcExtSys="10.193.219.120"
destExtSys="10.193.219.120"
srcSvc="sam_extXMLApi"
destSvc="central-mgr_dme"
response="yes">
    <outConfig>
        <tagDriver
  defDn="holder/tag-def-ep/type-Adapter Driver for HCR"
  defName="Adapter Driver for HCR"
  dn="holder/tag-ep/type-Adapter Driver for HCR-inst-[Cisco Ethernet
2.3.0.20]-obj-[compute/sys1/ch1/bl4/ad1]"
  objectName=""
  objectType="adaptorUnit"
  protocol="Ethernet"
  srcDme="resource-mgr"
  status="created"
  taggedObjectDn="compute/sys1/ch1/bl4/ad1"
  value="Cisco Ethernet 2.3.0.20"
  vendor="Cisco"
  version="2.3.0.20"/>
    </outConfig>
</configConfMo>
```

Example: Deleting an OS Tag

Request

```
<configConfMo
cookie="<real_cookie>"
 <inConfig>
<tagSoftwareInst dn="compute/sys1/chassis-1/blade-3" status='deleted' > //<dn of OsTag>
</tagSoftwareInst>
</inConfig>
</configConfMo>
```

Response

```
<configConfMo
dn=""
cookie="<real_cookie>"
commCookie="18/16/0/46b"
userContext="no"
srcExtSys="10.193.219.120"
destExtSys="10.193.219.120"
srcSvc="sam_extXMLApi"
destSvc="central-mgr_dme"
response="yes">
<outConfig>
</outConfig>
</configConfMo>
```

Example: Deleting an Adapter Tag

Request

```
<configConfMo
cookie="<real_cookie>"
 <inConfig>
<tagDriver dn='compute/sys1/chassis-1/blade-4/adaptor-1'> // <dn of DriverTag>
 status='deleted'
</ tagDriver >
</inConfig>
</configConfMo>
```

Response

```
<configConfMo
dn=""
cookie="<real_cookie>"
commCookie="18/16/0/46b7"
userContext="no"
srcExtSys="10.193.219.120"
destExtSys="10.193.219.120"
srcSvc="sam_extXMLApi"
destSvc="central-mgr_dme"
response="yes">
<outConfig>
</outConfig>
</configConfMo>
```

configConfMos

The configConfMos method configures managed objects in multiple subtrees using DNs.

Request Syntax

```
<xs:element name="configConfMos" type="configConfMos" substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfMos" mixed="true">
            <xs:all>
                <xs:element name="inConfigs" type="configMap" minOccurs="0">
                    <xs:unique name="unique_map_key_2">
                         <xs:selector xpath="pair"/>
                         <xs:field xpath="@key"/>
                    </xs:unique>
                </xs:element>
            </xs:all>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configConfMos" type="configConfMos" substitutionGroup="externalMethod"/>
        <xs:complexType name="configConfMos" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configMap" minOccurs="0">
                    <xs:unique name="unique_map_key_5">
                         <xs:selector xpath="pair"/>
                         <xs:field xpath="@key"/>
                    </xs:unique>
                </xs:element>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Configuring Managed Objects in Multiple Subtrees

Request

```
<configConfMos
    cookie="<real_cookie>">
    <inConfigs>
       <pair key="org-root/logprof-default">
          <policyLogProfile dn="org-root/logprof-default"
          name="default"
          level="debug1"
          size="10000000"
          backupCount="4"/>
       </pair>

 <!-- Update Controller Device Profile -->
       <pair key="org-root/controller-profile-default">
            <policyControllerDeviceProfile
               dn="org-root/controller-profile-default"
               adminState="enabled">
               .
                <commDnsProvider hostip="171.70.168.183" order="1"/>
                <commDnsProvider hostip="171.68.226.120" order="2"/>
                <commDnsProvider hostip="64.102.6.247"   order="3"/>
            </policyControllerDeviceProfile>
       </pair>
    </inConfigs>
</configConfMos>
```

Response

```
<configConfMos
    cookie="<real_cookie>"
    commCookie="7/15/0/1a74"
    srcExtSys="10.193.34.70"
    destExtSys="10.193.34.70"
    srcSvc="sam_extXMLApi"
    destSvc="policy-mgr_dme"
    response="yes">
    <outConfigs>
       <pair key="org-root/logprof-default">
          <policyLogProfile
             adminState="enabled"
             backupCount="4"
```

```
            descr="the log level for every process"
            dn="org-root/logprof-default"
            intId="10065"
            level="debug1"
            name="default"
            size="10000000"/>
        </pair>
        <pair key="org-root/controller-profile-default">
            .
            ./>
        </pair>
    </outConfigs>
</configConfMos>
```

configUCEstimateImpact

The configUCEstimateImpact method estimates the impact of a set of managed objects modifications in terms of disruption of running services. For example, modifying the UUID pool used by an updating template might require rebooting servers associated to service profiles instantiated from the template.

The user can estimate the impact of a change set by passing the set to the method and checking the corresponding ImpactAnalyzer object. The output parameter is the DN of a corresponding ImpactAnalyzer object for the change that the user passed down. The user use the configMoChangeEvent of the ImpactAnalyzer object. Once the state of this ImpactAnalyzer object is 'complete', the user can resolve the ImpactAnalyzer object to estimate impact results.

Estimate results contain the following information:

- Whether the changes are disruptive (for example, if the action requires a reboot of the server associated to the service profile).
- Whether the changes result in configuration issues; and the type of configuration issues.
- Number of UCS domains analyzed.
- Number of UCS domains affected.
- Number of suspended UCS domains.
- Number of UCS domains that lose visibility.
- Number of UCS domains that time out.
- Number of servers affected.
- Summary of changes.
- Time when the changes are applied (For example, immediately, after user acknowledgment, or during the scheduled occurrence of a maintenance window).

The parameters are defined as:

- configs—Set of changes to be evaluated (add, delete, or modify managed objects).
- ImpactAnalyzerID (optional)
- outImpactAnalyzerDN—DN of the corresponding ImpactAnalyzer object.

Request Syntax

```
<xs:element name="configUCEstimateImpact" type="configUCEstimateImpact"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configUCEstimateImpact" mixed="true">
            <xs:all>
                <xs:element name="inConfigs" type="configMap" minOccurs="0">
                    <xs:unique name="unique_map_key_3">
                         <xs:selector xpath="pair"/>
                         <xs:field xpath="@key"/>
                    </xs:unique>
                </xs:element>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configUCEstimateImpact" type="configUCEstimateImpact"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configUCEstimateImpact" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name=" ImpactAnalyzerDn " type=" referenceObject "/>
        </xs:complexType>
```

Example: Estimating the Impact of a Modification

Request

```
<configUCEstimateImpact
cookie="<real_cookie>"
inImpactAnalyzerId="0">
   <inConfigs>
<pair key="org-root/org-orgs/ls-gsp">
   <lsServer
   biosProfileName="global-SRIOV"
   dn="org-root/org-orgs/ls-gsp"
   status="modified"/>
</pair>
   </inConfigs>
</configUCEstimateImpact>
```

Response

```
<configUCEstimateImpact
cookie="<real_cookie>"
response="yes"
errorCode="0"
errorDescr=""
outImpactAnalyzerDn="impactanalyzer-ep/impact-analyzer-1401723100">
</configUCEstimateImpact>
```

configFindDependencies

The configFindDependencies method returns the device policy details for a specified policy.

Request Syntax

```
<xs:element name="configFindDependencies" type="configFindDependencies"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configFindDependencies" mixed="true">
            <xs:attribute name="inReturnConfigs">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configFindDependencies" type="configFindDependencies"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configFindDependencies" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="outHasDep">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example: Finding Device Policy Details

Request

```
<configFindDependencies
```

```
    dn="org-root/fw-host-pack-host-pack-6625"
    cookie="<real_cookie>"
    inReturnConfigs="yes">
</configFindDependencies>
```

Response

```
<configFindDependencies
    dn="org-root/fw-host-pack-host-pack-6625"
    cookie="<real_cookie>"
    response="yes"
    errorCode="0"
    errorDescr=""
    outHasDep="yes">
    <outConfigs>
       <lsServer
          agentPolicyName=""
          assignState="assigned"
          assocState="associated"
          biosProfileName=""
          bootPolicyName=""
          configQualifier=""
          configState="applied"
          descr=""
          dn="org-root/ls-service-profile-5"
          dynamicConPolicyName=""
          extIPState="none"
          fltAggr="0"
          fsmDescr=""
          fsmFlags=""
          fsmPrev="ConfigureSuccess"
          fsmProgr="100"
          fsmRmtInvErrCode="none"
          fsmRmtInvErrDescr=""
          fsmRmtInvRslt=""
          fsmStageDescr=""
          fsmStamp="2011-01-10T23:51:28.310"
          fsmStatus="nop"
          fsmTry="0"
          hostFwPolicyName="host-pack-6625"
          identPoolName=""
          intId="29191" localDiskPolicyName=""
          maintPolicyName=""
          mgmtAccessPolicyName=""
          mgmtFwPolicyName="m-firmware-1"
          name="service-profile-5"
          operBiosProfileName=""
          operBootPolicyName="org-root/boot-policy-default"
          operDynamicConPolicyName=""
          operHostFwPolicyName="org-root/fw-host-pack-host-pack-6625"
          operIdentPoolName="org-root/uuid-pool-default"
          operLocalDiskPolicyName="org-root/local-disk-config-default"
          operMaintPolicyName="org-root/maint-default"
          operMgmtAccessPolicyName=""
          operMgmtFwPolicyName="org-root/fw-mgmt-pack-m-firmware-1"
          operPowerPolicyName="org-root/power-policy-default"
          operScrubPolicyName="org-root/scrub-default"
          operSolPolicyName=""
          operSrcTemplName=""
          operState="ok"
          operStatsPolicyName="org-root/thr-policy-default"
          operVconProfileName=""
          owner="management"
          pnDn="sys/chassis-1/blade-5"
          powerPolicyName="default"
          scrubPolicyName=""
          solPolicyName=""
          srcTemplName=""
          statsPolicyName="default"
          type="instance"
```

```
          usrLbl=""
          uuid="derived"
          uuidSuffix="0000-000000000000"
          vconProfileName=""/>
    </outConfigs>
</configFindDependencies>
```

configMoChangeEvent

The configMoChangeEvent method provides event details from Cisco UCS Central as a result of event subscription. The status property indicates the action that caused the event (indicated by inEid) to be generated. This is a request sent from Cisco UCS Central to the subscribers. There is no response.

Request Syntax

```
<xs:element name="configMoChangeEvent" type="configMoChangeEvent"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configMoChangeEvent" mixed="true">
            <xs:all>
                <xs:element name="inConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inEid" type="xs:unsignedLong"/>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="configMoChangeEvent" type="configMoChangeEvent"
substitutionGroup="externalMethod"/>
        <xs:complexType name="configMoChangeEvent" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Providing Event Details to Subscribers

Request

```
<configMoChangeEvent
    cookie="<real_cookie>"
    inEid="174712">
    <inConfig>
       <callhomeEp
          dn="call-home"
          fsmPrev="configCallhomeSetLocal"
          fsmStamp="2008-10-16T17:59:25"
          fsmTry="11"
          status="modified"/>
    </inConfig>
</configMoChangeEvent>

<configMoChangeEvent
    cookie="<real_cookie>"
    inEid="174713">
```

```
    <inConfig>
       <mgmtIf
          dn="sys/switch-A/mgmt/if-1"
          fsmPrev="SwMgmtOobIfConfigSwitch"
          fsmStamp="2008-10-16T17:59:25"
          fsmTry="9"
          status="modified"/>
    </inConfig>
</configMoChangeEvent>

<configMoChangeEvent
    cookie="<real_cookie>"
    inEid="174714">
    <inConfig>
       <eventRecord
          affected="sys/sysdebug/file-export"
          cause="transition"
          created="2008-10-16T17:59:25"
          descr="[FSM:STAGE:RETRY:8]: configuring automatic core file export service on
            local"
          dn="event-log/54344"
          id="54344"
          ind="state-transition"
          severity="info"
          status="created"
          trig="special"
          txId="24839"
          user="internal"/>
    </inConfig>
</configMoChangeEvent>
```

Response

There is no response to this method.

configResolveChildren

The configResolveChildren method retrieves children of managed objects under a specific DN in the managed information tree. A filter can be used to reduce the number of children being returned.

Request Syntax

```
<xs:element name="configResolveChildren" type="configResolveChildren"
substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveChildren" mixed="true">
    <xs:all>
        <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="inDn" type="referenceObject"/>

    <xs:attribute name="inHierarchical">
        <xs:simpleType>
            <xs:union memberTypes="xs:boolean">
                 <xs:simpleType>
                     <xs:restriction base="xs:string">
                         <xs:enumeration value="no"/>
                         <xs:enumeration value="yes"/>
                     </xs:restriction>
                 </xs:simpleType>
            </xs:union>
        </xs:simpleType>
    </xs:attribute>
```

```
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>

    <xs:attribute name="classId" type="namingClassId"/>

</xs:complexType>
```

Response Syntax

```
<xs:element name="configResolveChildren" type="configResolveChildren"
substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveChildren" mixed="true">
    <xs:all>
        <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>

    <xs:attribute name="classId" type="namingClassId"/>

</xs:complexType>
```

Example: Computing the Scrub Policy Effect

Request

```
<configResolveChildren
cookie="<real_cookie>""

classId="computeScrubPolicy"
inDn="org-root"
inHierarchical="false">
    <inFilter>
    </inFilter>
</configResolveChildren>
```

Response

```
<configResolveChildren
cookie="<real_cookie>""
commCookie="7/16/0/25c"
srcExtSys="10.193.219.18"
destExtSys="10.193.219.18"
srcSvc="sam_extXMLApi"
destSvc="central-mgr_dme"
response="yes"
classId="computeScrubPolicy">
  <outConfigs>

<computeScrubPolicy
biosSettingsScrub="no"
descr=""
diskScrub="no"
dn="org-root/scrub-global-default"
flexFlashScrub="no"
intId="10238"
```

```
name="global-default"
policyLevel="0"
policyOwner="local"/>
  </outConfigs>
</configResolveChildren>
```

configResolveClass

The configResolveClass method returns requested managed object in a given class. If inHierarchical=true, the results contain children.

Request Syntax

```
<xs:element name="configResolveClass" type="configResolveClass"
substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveClass" mixed="true">
    <xs:all>
        <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="inHierarchical">
        <xs:simpleType>
            <xs:union memberTypes="xs:boolean">
                 <xs:simpleType>
                     <xs:restriction base="xs:string">
                         <xs:enumeration value="no"/>
                         <xs:enumeration value="yes"/>
                     </xs:restriction>
                 </xs:simpleType>
            </xs:union>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>

    <xs:attribute name="classId" type="namingClassId"/>

</xs:complexType>
```

Response Syntax

```
<xs:element name="configResolveClass" type="configResolveClass"
substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveClass" mixed="true">
    <xs:all>
        <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>

    <xs:attribute name="classId" type="namingClassId"/>

</xs:complexType>
```

Example: Searching for Organization Information

Request

```
<configResolveClass
cookie="<real_cookie>"

classId="orgOrg"
inHierarchical="false">
    <inFilter>
<eq class="orgOrg" property="level" value="root"/>
    </inFilter>
</configResolveClass>
```

Response

```
<configResolveClass
cookie="<real_cookie>"
commCookie="7/16/0/259"
srcExtSys="10.193.219.18"
destExtSys="10.193.219.18"
srcSvc="sam_extXMLApi"
destSvc="policy-mgr_dme"
response="yes"
classId="orgOrg">
  <outConfigs>

<orgOrg
descr=""
dn="org-root"
fltAggr="0"
level="root"
name="root"/>
  </outConfigs>
</configResolveClass>
```

configResolveClassIdx

The configResolveClass method returns requested managed object in a given class.

Request Syntax

```
        <xs:element name="configResolveClassIdx" type="configResolveClassIdx"
substitutionGroup="externalMethod"/>

        <xs:complexType name="configResolveClassIdx" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>

            <xs:attribute name="inQuery">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:minLength value="0"/>
                         <xs:maxLength value="510"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>

            <xs:attribute name="inClass">
```

```
            <xs:simpleType>
                <xs:restriction base="xs:string">
                     <xs:minLength value="0"/>
                     <xs:maxLength value="510"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>

        <xs:attribute name="inParentDn">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                     <xs:minLength value="0"/>
                     <xs:maxLength value="510"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>

        <xs:attribute name="inSortStr">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                     <xs:minLength value="0"/>
                     <xs:maxLength value="510"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>

        <xs:attribute name="inIncludeProp">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                     <xs:minLength value="0"/>
                     <xs:maxLength value="510"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>

        <xs:attribute name="inHierarchical">
            <xs:simpleType>
                <xs:union memberTypes="xs:boolean">
                     <xs:simpleType>
                         <xs:restriction base="xs:string">
                             <xs:enumeration value="no"/>
                             <xs:enumeration value="yes"/>
                         </xs:restriction>
                     </xs:simpleType>
                </xs:union>
            </xs:simpleType>
        </xs:attribute>

        <xs:attribute name="inOffset" type="xs:unsignedShort"/>

        <xs:attribute name="inLimit" type="xs:unsignedShort"/>

        <xs:attribute name="cookie" type="xs:string"/>
        <xs:attribute name="response" type="YesOrNo"/>

        <xs:attribute name="classId" type="namingClassId"/>

    </xs:complexType>
```

Response

```
<xs:element
name="configResolveClassIdx" type="configResolveClassIdx" substitutionGroup="externalMethod"/>
<xs:complexType name="configResolveClassIdx" mixed="true">
<xs:all>

<xs:element name="outConfigs" type="configSet" minOccurs="0"/>
</xs:all>

<xs:attribute name="outTotalCount" type="xs:unsignedInt"/>
```

```
<xs:attribute name="cookie" type="xs:string"/>
<xs:attribute name="response" type="YesOrNo"/>
<xs:attribute name="errorCode" type="xs:unsignedInt"/>
<xs:attribute name="errorDescr" type="xs:string"/>
<xs:attribute name="invocationResult" type="xs:string"/>
<xs:attribute name="classId" type="namingClassId"/>
</xs:complexType>
```

Example: Verifying the OS Vendor and Version

Request

```
<configResolveClassIdx
cookie='<real_cookie>'
inQuery='*:*'
inClass='hcOsInfoItem'
inSortStr='version+asc'
inOffset='0'
inLimit='100000'
/>
```

Response

```
<configResolveClassIdx outTotalCount="141">
    <outConfigs>
        <hcOsInfoItem vendor="CentOS" id="807" version="CentOS 6.4"/>
        <hcOsInfoItem vendor="CentOS" id="802" version="CentOS 6.5"/>
        <hcOsInfoItem vendor="CentOS" id="803" version="CentOS 6.6"/>
        <hcOsInfoItem vendor="CentOS" id="804" version="CentOS 6.7"/>
        <hcOsInfoItem vendor="CentOS" id="806" version="CentOS 7.0"/>
        <hcOsInfoItem vendor="CentOS" id="801" version="CentOS 7.1"/>
        <hcOsInfoItem vendor="CentOS" id="805" version="CentOS 7.2"/>
        <hcOsInfoItem vendor="Oracle" id="202" version="OL 7.0 UEK R3 U3"/>
        <hcOsInfoItem vendor="Oracle" id="2024" version="OL 7.1 UEK R3 U6"/>
        <hcOsInfoItem vendor="Oracle" id="2030" version="OL 7.1 UEK R3 U7"/>
        <hcOsInfoItem vendor="Oracle" id="2022" version="OL 7.1 UEK R4"/>
        <hcOsInfoItem vendor="Oracle" id="2023" version="OL 7.2 UEK R4"/>
        <hcOsInfoItem vendor="Oracle" id="2032" version="OL 7.2 UEK R4"/>
        <hcOsInfoItem vendor="Oracle" id="207" version="OVM 3.3.2"/>
        <hcOsInfoItem vendor="Oracle" id="2011" version="OVM 3.3.3"/>
        <hcOsInfoItem vendor="Oracle" id="2037" version="OVM 3.3.4"/>
        <hcOsInfoItem vendor="Oracle" id="2025" version="OVM 3.4"/>
        <hcOsInfoItem vendor="Red Hat" id="3018" version="Red Hat Enterprise Linux 6.0"/>
        <hcOsInfoItem vendor="Red Hat" id="3017" version="Red Hat Enterprise Linux 6.1"/>
        <hcOsInfoItem vendor="Red Hat" id="3011" version="Red Hat Enterprise Linux 6.2"/>
        <hcOsInfoItem vendor="Red Hat" id="308" version="Red Hat Enterprise Linux 6.3"/>
        <hcOsInfoItem vendor="Red Hat" id="302" version="Red Hat Enterprise Linux 6.4"/>
        <hcOsInfoItem vendor="Red Hat" id="301" version="Red Hat Enterprise Linux 6.5"/>
        <hcOsInfoItem vendor="Red Hat" id="305" version="Red Hat Enterprise Linux 6.6"/>
        <hcOsInfoItem vendor="Red Hat" id="3010" version="Red Hat Enterprise Linux 6.7"/>
        <hcOsInfoItem vendor="Red Hat" id="306" version="Red Hat Enterprise Linux 7.0"/>
        <hcOsInfoItem vendor="Red Hat" id="309" version="Red Hat Enterprise Linux 7.1"/>
        <hcOsInfoItem vendor="Red Hat" id="3013" version="Red Hat Enterprise Linux 7.2"/>
        <hcOsInfoItem vendor="SuSE" id="501" version="SUSE Linux Enterprise Server 11.3"/>
        <hcOsInfoItem vendor="SuSE" id="504" version="SUSE Linux Enterprise Server 11.4"/>
        <hcOsInfoItem vendor="SuSE" id="502" version="SUSE Linux Enterprise Server 12"/>
        <hcOsInfoItem vendor="SuSE" id="505" version="SUSE Linux Enterprise Server 12.1"/>
        <hcOsInfoItem vendor="Oracle" id="2043" version="Solaris 10 (U8)"/>
        <hcOsInfoItem vendor="Oracle" id="2026" version="Solaris 10 (U9)"/>
        <hcOsInfoItem vendor="Oracle" id="2029" version="Solaris 11"/>
        <hcOsInfoItem vendor="Oracle" id="2016" version="Solaris 11.1"/>
        <hcOsInfoItem vendor="Ubuntu" id="702" version="Ubuntu 14.04.2"/>
        <hcOsInfoItem vendor="Ubuntu" id="704" version="Ubuntu 14.04.3"/>
        <hcOsInfoItem vendor="Ubuntu" id="707" version="Ubuntu 14.04.4"/>
        <hcOsInfoItem vendor="Microsoft" id="406" version="Windows Server 2008"/>
        <hcOsInfoItem vendor="Microsoft" id="405" version="Windows Server 2008 R2"/>
```

```
        <hcOsInfoItem vendor="Microsoft" id="401" version="Windows Server 2008 R2 SP1"/>
        <hcOsInfoItem vendor="Microsoft" id="409" version="Windows Server 2008 SP1"/>
        <hcOsInfoItem vendor="Microsoft" id="404" version="Windows Server 2008 SP2"/>
        <hcOsInfoItem vendor="Microsoft" id="402" version="Windows Server 2012"/>
        <hcOsInfoItem vendor="Microsoft" id="403" version="Windows Server 2012 R2"/>
        <hcOsInfoItem vendor="Citrix" id="603" version="XenServer 6.1"/>
        <hcOsInfoItem vendor="Citrix" id="601" version="XenServer 6.2"/>
        <hcOsInfoItem vendor="Citrix" id="602" version="XenServer 6.5"/>
        <hcOsInfoItem vendor="Citrix" id="608" version="XenServer 6.5 SP1"/>
        <hcOsInfoItem vendor="VMware" id="1015" version="vSphere 5.1"/>
        <hcOsInfoItem vendor="VMware" id="106" version="vSphere 5.1 U1"/>
        <hcOsInfoItem vendor="VMware" id="102" version="vSphere 5.1 U2"/>
        <hcOsInfoItem vendor="VMware" id="104" version="vSphere 5.1 U3"/>
        <hcOsInfoItem vendor="VMware" id="101" version="vSphere 5.5"/>
        <hcOsInfoItem vendor="VMware" id="107" version="vSphere 5.5 U1"/>
        <hcOsInfoItem vendor="VMware" id="105" version="vSphere 5.5 U2"/>
        <hcOsInfoItem vendor="VMware" id="1011" version="vSphere 5.5 U3"/>
        <hcOsInfoItem vendor="VMware" id="1010" version="vSphere 6.0"/>
        <hcOsInfoItem vendor="VMware" id="1012" version="vSphere 6.0 U1"/>
        <hcOsInfoItem vendor="VMware" id="1017" version="vSphere 6.0 U2"/>
    </outConfigs>
</configResolveClassIdx>
```

Example: Verifying the Adapter Vendor and Version

Request

```
<configResolveClassIdx
cookie='<real_cookie>'
inQuery='*:*'
inClass='hcDriverInfoItem'
inSortStr='version+asc'
inOffset='0'
inLimit='100000'
/>
```

Response

```
<configResolveClassIdx outTotalCount="3940">
    <outConfigs>
        <hcDriverInfoItem adapterPid="N20-AC162-M4" vendor="Cisco" id="86000100203"
protocol="SNIC"
                          version="0.0.1.19"/>
        <hcDriverInfoItem adapterPid="N20-AC162-M4" vendor="Cisco" id="86000100202"
protocol="SNIC"
                          version="0.0.1.22"/>
        <hcDriverInfoItem adapterPid="N20-AC162-M4" vendor="Cisco" id="86000100201"
protocol="SNIC"
                          version="0.0.1.26"/>
        <hcDriverInfoItem adapterPid="UCS-SDHPCIE1600GB" vendor="Cisco" id="83000100203"
                          protocol="NVMe PCIe SSD"
                          version="0.8"/>
        <hcDriverInfoItem adapterPid="UCS-SDHPCIE800GB" vendor="Cisco" id="84000100203"
                          protocol="NVMe PCIe SSD"
                          version="0.8"/>
        <hcDriverInfoItem adapterPid="UCS-SDHPCIE1600GB" vendor="Cisco" id="83000100102"
protocol=" "
                          version="0.8 ()"/>
        <hcDriverInfoItem adapterPid="UCS-SDHPCIE800GB" vendor="Cisco" id="84000100102"
protocol=" "
                          version="0.8 ()"/>
        <hcDriverInfoItem adapterPid="UCS-SDHPCIE1600GB" vendor="Cisco" id="83000100202"
                          protocol="NVMe PCIe SSD"
                          version="0.9"/>
        <hcDriverInfoItem adapterPid="UCS-SDHPCIE800GB" vendor="Cisco" id="84000100202"
                          protocol="NVMe PCIe SSD"
```

```
                        version="0.9"/>
        <hcDriverInfoItem adapterPid="UCS-SDHPCIE1600GB" vendor="Cisco" id="83000100103"
protocol=" "
                        version="0.9 ()"/>
        <hcDriverInfoItem adapterPid="UCS-SDHPCIE800GB" vendor="Cisco" id="84000100103"
protocol=" "
                        version="0.9 ()"/>
        <hcDriverInfoItem adapterPid="R2XX-PL002" vendor="LSI Logic" id="850001001013"
protocol="RAID"
                        version="00.00.04.32.1vmw"/>
        <hcDriverInfoItem adapterPid="R2X0-ML002" vendor="LSI Logic" id="560001001013"
protocol="RAID"
                        version="00.00.05.30"/>
       <hcDriverInfoItem adapterPid="UCSC-RAID-SFFC200" vendor="LSI Logic" id="730001001023"

                        protocol="RAID"
                        version="00.00.05.30"/>
        <hcDriverInfoItem adapterPid="R2XX-PL002" vendor="LSI Logic" id="85000100101"
protocol="RAID"
                        version="00.00.05.30"/>
        <hcDriverInfoItem adapterPid="RC460-PL002" vendor="LSI Logic" id="260001001014"
protocol="RAID"
                        version="00.00.05.33"/>
        <hcDriverInfoItem adapterPid="UCS-RAID-9266" vendor="LSI Logic" id="100001001034"
protocol="RAID"
                        version="00.00.05.34"/>
       <hcDriverInfoItem adapterPid="UCS-RAID9286CV-8E" vendor="LSI Logic" id="180001001053"

                        protocol="RAID"
                        version="00.00.05.34"/>
    </outConfigs>
</configResolveClassIdx>
```

configResolveClasses

The configResolveClasses method returns requested managed objects in several classes. If inHierarchical=true, the results contain children.

Request Syntax

```
<xs:element name="configResolveClasses" type="configResolveClasses"
substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveClasses" mixed="true">
    <xs:all>
        <xs:element name="inIds" type="classIdSet" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="inHierarchical">
        <xs:simpleType>
            <xs:union memberTypes="xs:boolean">
                 <xs:simpleType>
                     <xs:restriction base="xs:string">
                         <xs:enumeration value="no"/>
                         <xs:enumeration value="yes"/>
                     </xs:restriction>
                 </xs:simpleType>
            </xs:union>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>

</xs:complexType>
```

Response Syntax

```
xs:element name="configResolveClasses" type="configResolveClasses"
substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveClasses" mixed="true">
    <xs:all>
        <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>

</xs:complexType>
```

Example: Resolving Hardware Information

Request

```
<configResolveClasses
cookie="<real_cookie>"
inHierarchical="false">
<inIds>
<Id value="equipmentChassis"/>
<Id value="computePhysical"/></inIds>
</configResolveClasses>
```

Response

```
<configResolveClasses
cookie="<real_cookie>"
commCookie="5/16/0/7f2"
srcExtSys="10.193.219.12"
destExtSys="10.193.219.12"
srcSvc="sam_extXMLApi"
destSvc="resource-mgr_dme"
response="yes">
  <outConfigs>

<equipmentChassis
adminState="acknowledged"
configState="ok"
connPath="A,B"
connStatus="A,B"
dn="compute/sys-1008/chassis-1"
.
./>

<equipmentChassis
adminState="acknowledged"
configState="ok"
connPath="A,B"
connStatus="A,B"
dn="compute/sys-1009/chassis-1"
.
./>

<computeRackUnit
adminPower="policy"
adminState="in-service"
```

```
assignedToDn=""
association="none"
availability="available"
availableMemory="16384"
checkPoint="discovered"
connPath="A,B"
connStatus="A,B"
descr=""
discovery="complete"
dn="compute/sys-1009/rack-unit-3"
.
./>

<computeRackUnit
adminPower="policy"
adminState="in-service"
.
./>
  </outConfigs>
</configResolveClasses>
```

configResolveDn

The configResolveDn method retrieves a single managed object for a specified DN.

Request Syntax

```
<xs:element name="configResolveDn" type="configResolveDn" substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveDn" mixed="true">
    <xs:attribute name="inHierarchical">
        <xs:simpleType>
            <xs:union memberTypes="xs:boolean">
                 <xs:simpleType>
                     <xs:restriction base="xs:string">
                         <xs:enumeration value="no"/>
                         <xs:enumeration value="yes"/>
                     </xs:restriction>
                 </xs:simpleType>
            </xs:union>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>

    <xs:attribute name="dn" type="referenceObject"/>
</xs:complexType>
```

Response Syntax

```
<xs:element name="configResolveDn" type="configResolveDn" substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveDn" mixed="true">
    <xs:all>
        <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>

    <xs:attribute name="dn" type="referenceObject"/>
```

```
</xs:complexType>
```

Example: Searching for Hardware

Request

```
<configResolveDn
cookie="<real_cookie>"
inHierarchical="false"
dn="compute/sys-1009/chassis-1/blade-1"
/>
```

Response

```
<configResolveDn
dn="compute/sys-1009/chassis-1/blade-1"
cookie="<real_cookie>"
commCookie="18/16/0/802"
srcExtSys="10.193.219.12"
destExtSys="10.193.219.12"
srcSvc="sam_extXMLApi"
destSvc="central-mgr_dme"
response="yes">
  <outConfig>
  <computeBlade
  adminPower="policy"
  adminState="in-service"
  assignedToDn=""
  association="none"
  availability="available"
  availableMemory="65536"
  chassisId="1"
  checkPoint="discovered"
  connPath="A,B"
  connStatus="A,B"
  descr=""
  discovery="complete"
  dn="compute/sys-1009/chassis-1/blade-1"
  .
  ./>
  </outConfig>
</configResolveDn>
```

configResolveDns

The configResolveDns method retrieves the managed objects for a list of DNs.

Request Syntax

```
<xs:element name="configResolveDns" type="configResolveDns"
substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveDns" mixed="true">
    <xs:all>
        <xs:element name="inDns" type="dnSet" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="inHierarchical">
```

```
        <xs:simpleType>
            <xs:union memberTypes="xs:boolean">
                 <xs:simpleType>
                     <xs:restriction base="xs:string">
                         <xs:enumeration value="no"/>
                         <xs:enumeration value="yes"/>
                     </xs:restriction>
                 </xs:simpleType>
            </xs:union>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>

</xs:complexType>
```

Response Syntax

```
<xs:element name="configResolveDns" type="configResolveDns"
substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveDns" mixed="true">
    <xs:all>
        <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
        <xs:element name="outUnresolved" type="dnSet" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>

</xs:complexType>
```

Example: Searching for Multiple Blades

Request

```
<configResolveDns
cookie="<real cookie>"
inHierarchical="false">
<inDns>
<dn value="compute/sys-1009/chassis-1/blade-1" />
<dn value="compute/sys-1009/chassis-1/blade-2" />
</inDns>
</configResolveDns>
```

Response

```
<configResolveDns
cookie="<real_cookie>"
commCookie="18/16/0/806"
srcExtSys="10.193.219.12"
destExtSys="10.193.219.12"
srcSvc="sam_extXMLApi"
destSvc="central-mgr_dme"
response="yes">
  <outConfigs>
```

```
<computeBlade
adminPower="policy"
adminState="in-service"
assignedToDn=""
association="none"
availability="available"
availableMemory="65536"
chassisId="1"
checkPoint="discovered"
connPath="A,B"
connStatus="A,B"
descr=""
discovery="complete"
dn="compute/sys-1009/chassis-1/blade-1"
.
./>

<computeBlade
adminPower="policy"
adminState="in-service"
assignedToDn=""
association="none"
availability="available"
availableMemory="65536"
chassisId="1"
checkPoint="discovered"
connPath="A,B"
connStatus="A,B"
descr=""
discovery="complete"
dn="compute/sys-1009/chassis-1/blade-2"
.
./>
  </outConfigs>
  <outUnresolved>
  </outUnresolved>
</configResolveDns>
```

configResolveParent

For a specified DN, the configResolveParent method retrieves the parent of the managed object.

Request Syntax

```
<xs:element name="configResolveParent" type="configResolveParent"
substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveParent" mixed="true">
    <xs:attribute name="inHierarchical">
        <xs:simpleType>
            <xs:union memberTypes="xs:boolean">
                 <xs:simpleType>
                     <xs:restriction base="xs:string">
                         <xs:enumeration value="no"/>
                         <xs:enumeration value="yes"/>
                     </xs:restriction>
                 </xs:simpleType>
            </xs:union>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>

    <xs:attribute name="dn" type="referenceObject"/>
</xs:complexType>
```

Response Syntax

```
<xs:element name="configResolveParent" type="configResolveParent"
substitutionGroup="externalMethod"/>

<xs:complexType name="configResolveParent" mixed="true">
    <xs:all>
        <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>

    <xs:attribute name="dn" type="referenceObject"/>

</xs:complexType>
```

Example: Locating a Blade

Request

```
<configResolveParent
cookie="<real_cookie>"
inHierarchical="false"
dn="compute/sys-1009/chassis-1/blade-1"/>
```

Response

```
<configResolveParent
dn="compute/sys-1009/chassis-1/blade-1"
cookie="<real_cookie>"
commCookie="18/16/0/809"
srcExtSys="10.193.219.12"
destExtSys="10.193.219.12"
srcSvc="sam_extXMLApi"
destSvc="central-mgr_dme"
response="yes">
  <outConfig>
  <equipmentChassis
  adminState="acknowledged"
  configState="ok"
  connPath="A,B"
  connStatus="A,B"
  dn="compute/sys-1009/chassis-1"
  .
  ./>
  </outConfig>
</configResolveParent>
```

configScope

The configScope method returns managed objects and details about their configuration.

Request Syntax

```
<xs:element name="configScope" type="configScope" substitutionGroup="externalMethod"/>

<xs:complexType name="configScope" mixed="true">
    <xs:all>
        <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="inClass" type="namingClassId"/>

    <xs:attribute name="inHierarchical">
        <xs:simpleType>
            <xs:union memberTypes="xs:boolean">
                 <xs:simpleType>
                     <xs:restriction base="xs:string">
                         <xs:enumeration value="no"/>
                         <xs:enumeration value="yes"/>
                     </xs:restriction>
                 </xs:simpleType>
            </xs:union>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="inRecursive">
        <xs:simpleType>
            <xs:union memberTypes="xs:boolean">
                 <xs:simpleType>
                     <xs:restriction base="xs:string">
                         <xs:enumeration value="no"/>
                         <xs:enumeration value="yes"/>
                     </xs:restriction>
                 </xs:simpleType>
            </xs:union>
        </xs:simpleType>
    </xs:attribute>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>

    <xs:attribute name="dn" type="referenceObject"/>

</xs:complexType>
```

Response Syntax

```
<xs:element name="configScope" type="configScope" substitutionGroup="externalMethod"/>

<xs:complexType name="configScope" mixed="true">
    <xs:all>
        <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>

    <xs:attribute name="dn" type="referenceObject"/>

</xs:complexType>
```

Example: Scoping a vNIC

Request

```
<configScope
dn="org-root/org-org-1/org-org-2"
cookie="<real_cookie>"
inClass="vnicLanConnPolicy"
inHierarchical="false"
inRecursive="false">
<inFilter>
</inFilter>
</configScope>
```

Response

```
<configScope
dn="org-root/org-org-1/org-org-2"
cookie="<real_cookie>"
commCookie="18/16/0/814"
srcExtSys="10.193.219.12"
destExtSys="10.193.219.12"
srcSvc="sam_extXMLApi"
destSvc="central-mgr_dme"
response="yes">
<outConfigs>

<vnicLanConnPolicy
descr=""
dn="org-root/org-org-1/org-org-2/lan-conn-pol-lcp-1"
fltAggr="0"
intId="13667"
name="lcp-1"
policyLevel="3"
policyOwner="policy"/>

<vnicLanConnPolicy
descr=""
dn="org-root/org-org-1/org-org-2/lan-conn-pol-lcp-2"
fltAggr="0"
intId="13668"
name="lcp-2"
policyLevel="3"
policyOwner="policy"/>
</outConfigs>
</configScope>
```

eventSubscribe

The eventSubscribe method allows a client to subscribe to asynchronous events generated by Cisco UCS Central, including all object changes in the system (created, changed, or deleted).

Event subscription allows a client application to register for event notification from Cisco UCS Central. When an event occurs, Cisco UCS Central informs the client application of the event and its type. Only the actual change information is sent. The object's unaffected attributes are not included.

Request Syntax

```
<xs:element name="eventSubscribe" type="eventSubscribe"
            substitutionGroup="externalMethod"/>
        <xs:complexType name="eventSubscribe" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="eventSubscribe" type="eventSubscribe"
substitutionGroup="externalMethod"/>
        <xs:complexType name="eventSubscribe" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Registering for an Event

Request

```
<eventSubscribe
    cookie="<real_cookie>"
    <inFilter>
    </inFilter>
</eventSubscribe>
```

Response

```
NO RESPONSE OR ACKNOWLEDGMENT.
```

eventUnsubscribe

The eventUnsubscribe method allows a client to unsubscribe from asynchronous events generated by Cisco UCS Central, reversing event subscriptions that resulted from eventSubscribe.

Request Syntax

```
<xs:element name="eventUnsubscribe" type="eventUnsubscribe"
            substitutionGroup="externalMethod"/>
        <xs:complexType name="eventUnsubscribe" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="eventUnsubscribe" type="eventUnsubscribe"
substitutionGroup="externalMethod"/>
        <xs:complexType name="eventUnsubscribe" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Unregistering for an Event

Request

```
<eventUnsubscribe
    cookie="<real_cookie>"
    <inFilter>
    </inFilter>
</eventUnsubscribe>
```

Response

```
NO RESPONSE OR ACKNOWLEDGMENT.
```

faultAckFaultByDn

The faultAckFaultByDn method acknowledges a fault using the DN as input. The acknowledgment response marks the fault severity as cleared. Faults categorized as auto-cleared do not require acknowledgment.

Request Syntax

```
<!--
   - Method: fault:AckFaultByDn
-->
<xs:element name="faultAckFaultByDn" type="faultAckFaultByDn"
substitutionGroup="externalMethod"/>

<xs:complexType name="faultAckFaultByDn" mixed="true">
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>

    <xs:attribute name="dn" type="referenceObject"/>

</xs:complexType>
```

Response Syntax

```
<!--
   - Method: fault:AckFaultByDn
-->
<xs:element name="faultAckFaultByDn" type="faultAckFaultByDn"
substitutionGroup="externalMethod"/>

<xs:complexType name="faultAckFaultByDn" mixed="true">
```

```
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
    <xs:attribute name="errorCode" type="xs:unsignedInt"/>
    <xs:attribute name="errorDescr" type="xs:string"/>
    <xs:attribute name="invocationResult" type="xs:string"/>

    <xs:attribute name="dn" type="referenceObject"/>

</xs:complexType>
```

Example: Acknowledging and Clearing a Fault

Request

```
<faultAckFaultByDn
cookie="<real_cookie>"
dn="sys/switch-A/stor-part-bootflash/fault-F10000336">
</faultAckFaultByDn>
```

Response

```
<faultAckFaultByDn
dn="sys/switch-A/stor-part-bootflash/fault-F10000336"
cookie="<real_cookie>"
commCookie="18/16/0/98e"
srcExtSys="10.193.219.18"
destExtSys="10.193.219.18"
srcSvc="sam_extXMLApi"
destSvc="central-mgr_dme"
response="yes">
</faultAckFaultByDn>
```

faultAckFaultsByDn

The faultAckFaultsByDn method acknowledges multiple faults using DN as the input. The acknowledgment response marks the fault severity as cleared. Faults categorized as auto-cleared do not require acknowledgment.

Request Syntax

```
<!--
   - Method: fault:AckFaultsByDn
-->
<xs:element name="faultAckFaultsByDn" type="faultAckFaultsByDn"
substitutionGroup="externalMethod"/>

<xs:complexType name="faultAckFaultsByDn" mixed="true">
    <xs:all>
        <xs:element name="inDns" type="dnSet" minOccurs="0"/>
    </xs:all>

    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>

</xs:complexType>
```

Response Syntax

```
<xs:element name="faultAckFaults" type="faultAckFaults"
```

```
substitutionGroup="externalMethod"/>
        <xs:complexType name="faultAckFaults" mixed="true">
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
        </xs:complexType>
```

Example: Acknowledging and Clearing Multiple Faults

Request

```
<faultAckFaultsByDn
cookie="<real_cookie>"
<inDns>
<dn
value="sys/corefiles/file-1401449743_SAM_kondal-vm-aub115_svc_centralMgr__log.11944.tar.gz/fault-F10000005"/>
<dn value="org-root/wwn-pool-global-node-default/fault-F10000200"/>
</inDns>
</faultAckFaultsByDn>
```

Response

```
<faultAckFaultsByDn
cookie="<real_cookie>"
commCookie="18/16/0/99c"
srcExtSys="10.193.219.18"
destExtSys="10.193.219.18"
srcSvc="sam_extXMLApi"
destSvc="central-mgr_dme"
response="yes">
</faultAckFaultsByDn>
```

lsClone

The lsClone method clones a service profile or a service profile template.

Request Syntax

```
<xs:element name="lsClone" type="lsClone" substitutionGroup="externalMethod"/>
        <xs:complexType name="lsClone" mixed="true">
            <xs:attribute name="inTargetOrg" type="referenceObject"/>
            <xs:attribute name="inServerName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
```

```
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="lsClone" type="lsClone" substitutionGroup="externalMethod"/>
        <xs:complexType name="lsClone" mixed="true">
            <xs:all>
                <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example: Cloning a Service Profile

Request

```
<lsClone
    dn="org-root/ls-SP1"
    cookie="<real_cookie>"
    inTargetOrg="org-root"
    inServerName="CP-1"
    inHierarchical="no">
</lsClone>
```

Response

```
<lsClone
    dn="org-root/ls-SP1"
    cookie="<real_cookie>"
    response="yes"
    errorCode="0"
    errorDescr="">
    <outConfig>
       <lsServer
          agentPolicyName=""
          assignState="unassigned"
          assocState="unassociated"
          biosProfileName=""
          bootPolicyName=""
          configQualifier=""
          configState="not-applied"
          descr=""
          dn="org-root/ls-CP-1"
          dynamicConPolicyName=""
          extIPState="none"
          fltAggr="0"
          hostFwPolicyName=""
          identPoolName="default"
          intId="52365"
          localDiskPolicyName="default"
```

```
            maintPolicyName=""
            mgmtAccessPolicyName=""
            mgmtFwPolicyName=""
            name="CP-1"
            operBiosProfileName=""
            operBootPolicyName=""
            operDynamicConPolicyName=""
            operHostFwPolicyName=""
            operIdentPoolName=""
            operLocalDiskPolicyName=""
            operMaintPolicyName=""
            operMgmtAccessPolicyName=""
            operMgmtFwPolicyName=""
            operPowerPolicyName=""
            operScrubPolicyName=""
            operSolPolicyName=""
            operSrcTemplName=""
            operState="unassociated"
            operStatsPolicyName=""
            operVconProfileName=""
            owner="management"
            pnDn=""
            powerPolicyName="default"
            scrubPolicyName=""
            solPolicyName=""
            srcTemplName="service-templ-001"
            statsPolicyName="default"
            status="created"
            type="instance"
            usrLbl=""
            uuid="derived"
            uuidSuffix="0000-000000000000"
            vconProfileName=""/>
    </outConfig>
</lsClone>
```

Example: Cloning a Service Profile Template

Request

```
<lsClone
    dn="org-root/ls-template-3"
    cookie="<real_cookie>"
    inTargetOrg="org-root"
    inServerName="CT-1"
    inHierarchical="no">
</lsClone>
```

Response

```
<lsClone
    dn="org-root/ls-template-3"
    cookie="<real_cookie>"
    response="yes"
    errorCode="0"
    errorDescr="">
    <outConfig>
       <lsServer
          agentPolicyName=""
          assignState="unassigned"
          assocState="unassociated"
          biosProfileName=""
          bootPolicyName=""
          configQualifier=""
```

```
            configState="not-applied"
            descr=""
            dn="org-root/ls-CT-1"
            dynamicConPolicyName=""
            extIPState="none"
            fltAggr="0"
            hostFwPolicyName=""
            identPoolName="default"
            intId="52389"
            localDiskPolicyName=""
            maintPolicyName=""
            mgmtAccessPolicyName=""
            mgmtFwPolicyName=""
            name="CT-1"
            operBiosProfileName=""
            operBootPolicyName=""
            operDynamicConPolicyName=""
            operHostFwPolicyName=""
            operIdentPoolName=""
            operLocalDiskPolicyName=""
            operMaintPolicyName=""
            operMgmtAccessPolicyName=""
            operMgmtFwPolicyName=""
            operPowerPolicyName=""
            operScrubPolicyName=""
            operSolPolicyName=""
            operSrcTemplName=""
            operState="unassociated"
            operStatsPolicyName=""
            operVconProfileName=""
            owner="management"
            pnDn=""
            powerPolicyName="default"
            scrubPolicyName=""
            solPolicyName=""
            srcTemplName=""
            statsPolicyName="default"
            status="created"
            type="updating-template"
            usrLbl=""
            uuid="derived"
            uuidSuffix="0000-000000000000"
            vconProfileName=""/>
    </outConfig>
</lsClone>
```

lsInstantiateNNamedTemplate

The lsInstantiateNNamedTemplate method takes the specified service profile template and creates the desired number of service profiles. This method uses the following parameters:

- dn—Specifies the service template used to create the new service profiles.
- nameSet—Contains the names of the service profiles to be created.
- targetOrg—Specifies the organization under which these service profiles are created.

Request Syntax

```
<xs:element name="lsInstantiateNNamedTemplate" type="lsInstantiateNNamedTemplate"
substitutionGroup="externalMethod"/>
        <xs:complexType name="lsInstantiateNNamedTemplate" mixed="true">
            <xs:all>
                <xs:element name="inNameSet" type="dnSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inTargetOrg" type="referenceObject"/>
```

```
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="lsInstantiateNNamedTemplate" type="lsInstantiateNNamedTemplate"
substitutionGroup="externalMethod"/>
        <xs:complexType name="lsInstantiateNNamedTemplate" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Creating a Service Profiles from a Specific Template

Request

```
<lsInstantiateNNamedTemplate
    dn="org-root/ls-service-template-001"
    cookie="<real_cookie>"
    inTargetOrg="org-root"
    inHierarchical="no">
    <inNameSet>
       <dn value="service-profile-A"/>
       <dn value="service-profile-B"/>
       <dn value="service-profile-C"/>
    </inNameSet>
</lsInstantiateNNamedTemplate>
```

Response

```
<lsInstantiateNNamedTemplate
    dn="org-root/ls-service-template-001"
    cookie="<real_cookie>"
    response="yes"
    errorCode="0"
    errorDescr="">
    <outConfigs>
       <lsServer
          agentPolicyName=""
          assignState="unassigned"
```

```
          assocState="unassociated"
          biosProfileName=""
          bootPolicyName=""
          configQualifier=""
          configState="not-applied"
          descr=""
          dn="org-root/ls-service-profile-A "
          dynamicConPolicyName=""
          .
          .
          status="created"
          type="instance"
          usrLbl=""
          uuid="derived"
          uuidSuffix="0000-000000000000"
          vconProfileName=""/>
       <lsServer
          .
       ./>
       <lsServer
          .
       ./>
    </outConfigs>
</lsInstantiateNNamedTemplate>
```

lsInstantiateNTemplate

The lsInstantiateNTemplate method creates a number (N) of service profiles from a template.

Request Syntax

```
<xs:element name="lsInstantiateNTemplate" type="lsInstantiateNTemplate"
substitutionGroup="externalMethod"/>
        <xs:complexType name="lsInstantiateNTemplate" mixed="true">
            <xs:attribute name="inTargetOrg" type="referenceObject"/>
            <xs:attribute name="inServerNamePrefixOrEmpty">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inNumberOf" type="xs:unsignedByte"/>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="lsInstantiateNTemplate" type="lsInstantiateNTemplate"
substitutionGroup="externalMethod"/>
        <xs:complexType name="lsInstantiateNTemplate" mixed="true">
```

```
            <xs:all>
                <xs:element name="outConfigs" type="configSet" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example: Creating Multiple Service Profiles from a Template

Request

```
<lsInstantiateNTemplate
    dn="org-root/ls-service-templ-001"
    cookie="<real_cookie>"
    inTargetOrg="org-root"
    inServerNamePrefixOrEmpty="SP"
    inNumberOf="2"
    inHierarchical="no">
</lsInstantiateNTemplate>
```

Response

```
<lsInstantiateNTemplate
    dn="org-root/ls-service-templ-001"
    cookie="<real_cookie>"
    response="yes"
    errorCode="0"
    errorDescr="">
    <outConfigs>
       <lsServer
          agentPolicyName=""
          assignState="unassigned"
          assocState="unassociated"
          biosProfileName=""
          bootPolicyName=""
          configQualifier=""
          configState="not-applied"
          descr=""
          dn="org-root/ls-SP1"
          dynamicConPolicyName=""
          extIPState="none"
          fltAggr="0"
          hostFwPolicyName=""
          identPoolName="default"
          intId="52227"
          localDiskPolicyName="default"
          maintPolicyName=""
          mgmtAccessPolicyName=""
          mgmtFwPolicyName=""
          name="SP1"
          operBiosProfileName=""
          operBootPolicyName=""
          operDynamicConPolicyName=""
          operHostFwPolicyName=""
          operIdentPoolName=""
          operLocalDiskPolicyName=""
          operMaintPolicyName=""
          operMgmtAccessPolicyName=""
          operMgmtFwPolicyName=""
          operPowerPolicyName=""
```

```
          operScrubPolicyName=""
          operSolPolicyName=""
          operSrcTemplName=""
          operState="unassociated"
          operStatsPolicyName=""
          operVconProfileName=""
          owner="management"
          pnDn=""
          powerPolicyName="default"
          scrubPolicyName=""
          solPolicyName=""
          srcTemplName="service-templ-001"
          statsPolicyName="default"
          status="created"
          type="instance"
          usrLbl=""
          uuid="derived"
          uuidSuffix="0000-000000000000"
          vconProfileName=""/>
       <lsServer
          .
      ./>
    </outConfigs>
</lsInstantiateNTemplate>
```

lsInstantiateTemplate

The lsInstantiateTemplate method creates one service profile from a specified template.

Request Syntax

```
<xs:element name="lsInstantiateTemplate" type="lsInstantiateTemplate"
substitutionGroup="externalMethod"/>
        <xs:complexType name="lsInstantiateTemplate" mixed="true">
            <xs:attribute name="inTargetOrg" type="referenceObject"/>

            <xs:attribute name="inServerName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="lsInstantiateTemplate" type="lsInstantiateTemplate"
substitutionGroup="externalMethod"/>
        <xs:complexType name="lsInstantiateTemplate" mixed="true">
```

```
            <xs:all>
                <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example: Creating a Service Profile from a Specific Template

Request

```
<lsInstantiateTemplate
    dn="org-root/ls-service-templ-001"
    cookie="<real_cookie>"
    inTargetOrg="org-root"
    inServerName="SP1"
    inHierarchical="no">
 </lsInstantiateTemplate>
```

Response

```
<lsInstantiateTemplate
    dn="org-root/ls-service-templ-001"
    cookie="<real_cookie>"
    response="yes"
    errorCode="0"
    errorDescr="">
    <outConfigs>
       <lsServer
          agentPolicyName=""
          assignState="unassigned"
          assocState="unassociated"
          biosProfileName=""
          bootPolicyName=""
          configQualifier=""
          configState="not-applied"
          descr=""
          dn="org-root/ls-SP1"
          dynamicConPolicyName=""
          extIPState="none"
          fltAggr="0"
          fsmDescr=""
          fsmFlags=""
          fsmPrev="nop"
          fsmProgr="100"
          fsmRmtInvErrCode="none"
          fsmRmtInvErrDescr=""
          fsmRmtInvRslt=""
          fsmStageDescr=""
          fsmStamp="never"
          fsmStatus="nop"
          fsmTry="0"
          hostFwPolicyName=""
          identPoolName="default"
          intId="52227"
          localDiskPolicyName="default"
          maintPolicyName=""
          mgmtAccessPolicyName=""
          mgmtFwPolicyName=""
          name="SP1"
```

```
            operBiosProfileName=""
            operBootPolicyName=""
            operDynamicConPolicyName=""
            operHostFwPolicyName=""
            operIdentPoolName=""
            operLocalDiskPolicyName=""
            operMaintPolicyName=""
            operMgmtAccessPolicyName=""
            operMgmtFwPolicyName=""
            operPowerPolicyName=""
            operScrubPolicyName=""
            operSolPolicyName=""
            operSrcTemplName=""
            operState="unassociated"
            operStatsPolicyName=""
            operVconProfileName=""
            owner="management"
            pnDn=""
            powerPolicyName="default"
            scrubPolicyName=""
            solPolicyName=""
            srcTemplName="service-templ-001"
            statsPolicyName="default"
            status="created"
            type="instance"
            usrLbl=""
            uuid="derived"
            uuidSuffix="0000-000000000000"
            vconProfileName=""/>
    </outConfigs>
</lsInstantiateTemplate>
```

lsTemplatise

The lsTemplatise method creates a template from a specified service profile.

Request Syntax

```
<xs:element name="lsTemplatise" type="lsTemplatise" substitutionGroup="externalMethod"/>
        <xs:complexType name="lsTemplatise" mixed="true">
            <xs:attribute name="inTargetOrg" type="referenceObject"/>

            <xs:attribute name="inTemplateName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:pattern value="[\-\.:_a-zA-Z0-9]{0,16}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inTemplateType">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                         <xs:enumeration value="instance"/>
                         <xs:enumeration value="initial-template"/>
                         <xs:enumeration value="updating-template"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
```

```
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="lsTemplatise" type="lsTemplatise" substitutionGroup="externalMethod"/>
        <xs:complexType name="lsTemplatise" mixed="true">
            <xs:all>
                <xs:element name="outConfig" type="configConfig" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example: Creating a Template from a Specific Service Profile

Request

```
<lsTemplatise
    dn="org-root/ls-SP1"
    cookie="<real_cookie>"
    inTargetOrg="org-root"
    inTemplateName="tempate-2"
    inTemplateType="initial-template"
    inHierarchical="no">
</lsTemplatise>
```

Response

```
<lsTemplatise
    dn="org-root/ls-SP1"
    cookie="<real_cookie>"
    response="yes"
    errorCode="0"
    errorDescr="">
    <outConfig>
       <lsServer
          agentPolicyName=""
          assignState="unassigned"
          assocState="unassociated"
          biosProfileName=""
          bootPolicyName=""
          configQualifier=""
          configState="not-applied"
          descr=""
          dn="org-root/ls-tempate-2"
          dynamicConPolicyName=""
          extIPState="none"
          fltAggr="0"
          hostFwPolicyName=""
          identPoolName="default"
          intId="52339"
          localDiskPolicyName="default"
```

```
            maintPolicyName=""
            mgmtAccessPolicyName=""
            mgmtFwPolicyName=""
            name="tempate-2"
            operBiosProfileName=""
            operBootPolicyName=""
            operDynamicConPolicyName=""
            operHostFwPolicyName=""
            operIdentPoolName=""
            operLocalDiskPolicyName=""
            operMaintPolicyName=""
            operMgmtAccessPolicyName=""
            operMgmtFwPolicyName=""
            operPowerPolicyName=""
            operScrubPolicyName=""
            operSolPolicyName=""
            operSrcTemplName=""
            operState="unassociated"
            operStatsPolicyName=""
            operVconProfileName=""
            owner="management"
            pnDn=""
            powerPolicyName="default"
            scrubPolicyName=""
            solPolicyName=""
            srcTemplName="service-templ-001"
            statsPolicyName="default"
            status="created"
            type="initial-template"
            usrLbl=""
            uuid="derived"
            uuidSuffix="0000-000000000000"
            vconProfileName=""/>
    </outConfig>
</lsTemplatise>
```

poolResolveInScope

The poolResolveInScope method, using the specified DN, looks up the pool and parent pools (optional) recursively to the root. If no pool exists, an empty map is returned. If any pool is found, this method searches all pools with the specified class and filters.

Note If inSingleLevel = false, this method searches parent pools up to the root directory.

Request Syntax

```
<xs:element name="poolResolveInScope" type="poolResolveInScope"
substitutionGroup="externalMethod"/>
        <xs:complexType name="poolResolveInScope" mixed="true">
            <xs:all>
                <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
            </xs:all>
            <xs:attribute name="inClass" type="namingClassId"/>
            <xs:attribute name="inSingleLevel">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
```

```
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="inHierarchical">
                <xs:simpleType>
                    <xs:union memberTypes="xs:boolean">
                         <xs:simpleType>
                             <xs:restriction base="xs:string">
                                 <xs:enumeration value="no"/>
                                 <xs:enumeration value="yes"/>
                             </xs:restriction>
                         </xs:simpleType>
                    </xs:union>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Response Syntax

```
<xs:element name="poolResolveInScope" type="poolResolveInScope"
substitutionGroup="externalMethod"/>
        <xs:complexType name="poolResolveInScope" mixed="true">
            <xs:all>
                <xs:element name="outConfigs" type="configMap" minOccurs="0">
                    <xs:unique name="unique_map_key_13">
                         <xs:selector xpath="pair"/>
                         <xs:field xpath="@key"/>
                    </xs:unique>
                </xs:element>
            </xs:all>
            <xs:attribute name="cookie" type="xs:string"/>
            <xs:attribute name="response" type="YesOrNo"/>
            <xs:attribute name="errorCode" type="xs:unsignedInt"/>
            <xs:attribute name="errorDescr" type="xs:string"/>
            <xs:attribute name="invocationResult" type="xs:string"/>
            <xs:attribute name="dn" type="referenceObject"/>
        </xs:complexType>
```

Example: Searching for Parent Directories of a Pool

Request

```
<poolResolveInScope
    dn="org-root/org-Cisco"
    cookie="<real_cookie>"
    class=fwPool />
```

Response

```
<poolResolveInScope
    dn="org-root/org-Cisco"
    cookie="<real_cookie>"
    commCookie="5/15/0/5bf"
    srcExtSys="10.193.33.221"
    destExtSys="10.193.33.221"
    srcSvc="sam_extXMLApi"
    destSvc="resource-mgr_dme"
    response="yes">
    <outConfigs>
```

```
        <pair key="fwpool-default">
           <fwPool
              assigned="0"
              descr="Default Pool of Virtual Security Gateway resources"
              dn="org-root/fwpool-default"
              fltAggr="65536"
              id="1"
              intId="10065"
              name="default"
              size="0"/>
        </pair>
        <pair key="fwpool-ciscoCfwPool">
           .
        </pair>
    </outConfigs>
</poolResolveInScope>
```

CHAPTER 4

Cisco UCS Central XML Object Access Privileges

This chapter includes the following sections:

Privileges Summary Table

When users are assigned to a role, that role allows certain privileges. Those privileges allow the user access to specific system resources and authorize permission to perform tasks on those resources. The following table lists each privilege and the initial default user role that has been given that privilege.

Internal Name	Label	Description	Default Role Assignment
aaa, on page 89	AAA	System security and AAA	AAA Administrator
admin, on page 90	ADMIN	Access to everything (combines all roles)	Administrator
ext-lan-config, on page 90	EXT_LAN_CONFIG	Configuration of network end points, UCDs, etc.	Network Administrator
ext-lan-policy, on page 91	EXT_LAN_POLICY	External network policies	Network Administrator
ext-lan-qos, on page 91	EXT_LAN_QOS	External LAN QoS	Network Administrator
ext-lan-security, on page 91	EXT_LAN_SECURITY	External LAN security	Network Administrator
ext-san-config, on page 92	EXT_SAN_CONFIG	Configuration of network end points, UCDs, etc.	Storage Administrator

Internal Name	Label	Description	Default Role Assignment
ext-san-policy, on page 92	EXT_SAN_POLICY	External SAN policy	Storage Administrator
ext-san-qos, on page 92	EXT_SAN_QOS	External SAN QoS	Storage Administrator
ext-san-security, on page 93	EXT_SAN_SECURITY	External SAN security (VACLs, etc.)	Storage Administrator
fault, on page 93	FAULT	Alarms, alarm policies, etc.	Operations
ls-config, on page 93	LS_CONFIG	Service profile configuration	Server Profile Administrator
ls-config-policy, on page 94	LS_CONFIG_POLICY	Service profile configuration policy	Server Profile Administrator
ls-ext-access, on page 94	LS_EXT_ACCESS	Service profile end point access	Server Profile Administrator
ls-network, on page 94	LS_NETWORK	Service profile network	Network Administrator
ls-network-policy, on page 95	LS_NETWORK_POLICY	Setting up MAC pools, etc.	Network Administrator
ls-power, on page 95	LS_POWER	LS power management	Facility Manager
ls-qos, on page 95	LS_QOS	Service profile QoS	Network Administrator
ls-qos-policy, on page 96	LS_QOS_POLICY	Setting up ls-level QoS	Network Administrator
ls-security, on page 96	LS_SECURITY	Service profile security	Server Security Administrator
ls-security-policy, on page 96	LS_SECURITY_POLICY	Setting up security policies	Server Security Administrator
ls-storage, on page 98	LS_STORAGE	Service profile storage	Storage Administrator
ls-storage-policy, on page 98	LS_STORAGE_POLICY	Service profile storage policy	Storage Administrator

Internal Name	Label	Description	Default Role Assignment
operations, on page 99	OPERATIONS	Logs, call home functionality, etc.	Operations
pn-equipment, on page 100	PN_EQUIPMENT	Server hardware management	Server Equipment Administrator
pn-maintenance, on page 100	PN_MAINTENANCE	Server maintenance (update BIOS, etc.)	Server Equipment Administrator
pn-policy, on page 100	PN_POLICY	Physical server policies	Server Equipment Administrator
pn-security, on page 101	PN_SECURITY	Physical node security	Server Security Administrator
pod-config, on page 101	POD_CONFIG	Pod configuration	Network Administrator
pod-policy, on page 101	POD_POLICY	Pod policies	Network Administrator
pod-qos, on page 102	POD_QOS	Internal pod-QoS (if needed)	Network Administrator
pod-security, on page 102	POD_SECURITY	Pod security	Network Administrator
power-mgmt, on page 102	POWER_MGMT	Data center power management	Facility Manager
read-only, on page 102	READ_ONLY	Read-only access	Available to all roles

Privileges

aaa

Purpose

System security and AAA.

This privilege has read and write access to all users, roles, AAA, and communication services configuration. Read access is available for all other objects.

Responsible Role

AAA Administrator

Controlled Objects

aaa:AuthRealm, aaa:EpAuthProfile, aaa:EpUser, aaa:ExtMgmtCutThruTkn, aaa:LdapEp, aaa:LdapProvider, aaa:Locale, aaa:Log, aaa:Org, aaa:RadiusEp, aaa:RadiusProvider, aaa:RemoteUser, aaa:Role, aaa:Session, aaa:SshAuth, aaa:TacacsPlusEp, aaa:TacacsPlusProvider, aaa:User, aaa:UserEp, aaa:UserLocale, aaa:UserRole, comm:Cimxml, comm:Dns, comm:DnsProvider, comm:EvtChannel, comm:Http, comm:Https, comm:SmashCLP, comm:Snmp, comm:SnmpTrap, comm:SnmpUser, comm:Ssh, comm:SvcEp, comm:Telnet, comm:WebChannel, comm:Wsman, comm:XmlClConnPolicy, comm:XmlClConnPolicy, pki:CertReq, pki:KeyRing, pki:TP

admin

Purpose

System administration

Responsible Role

Administrator

Controlled Objects

This role is system level. The administrator controls all objects.

ext-lan-config

Purpose

External LAN configuration

Responsible Role

Network Administrator

Controlled Objects

adaptor:ExtIf, adaptor:ExtEthIf, adaptor:HostIf, adaptor:HostEthIf, adaptor:HostFcIf, comm:DateTime, comm:Dns, comm:DnsProvider, comm:NtpProvider, fabric:EthLan, fabric:EthLanEp, fabric:EthLanPc, fabric:EthLanPcEp, fabric:LanCloud, fabric:LanPinGroup, fabric:LanPinTarget, fabric:Vlan, macpool:Format, network:Element, top:System, vnic:FcOEIf, vnic:LanConnTempl

ext-lan-policy

Purpose

External LAN policy

Responsible Role

Network Administrator

Controlled Objects

adaptor:ExtIf, adaptor:ExtEthIf, adaptor:HostIf, adaptor:HostEthIf, adaptor:HostFcIf, fabric:EthLan, fabric:EthLanEp, fabric:EthLanPc, fabric:EthLanPcEp, fabric:LanCloud, fabric:LanPinGroup, fabric:LanPinTarget, fabric:VCon, fabric:VConProfile, fabric:Vlan, macpool:Format, vnic:FcOEIf, vnic:LanConnTempl

ext-lan-qos

Purpose

External LAN QoS

Responsible Role

Network Administrator

Controlled Objects

qosclass:Definition, qosclass:EthBE, qosclass:EthClassified, qosclass:Fc

ext-lan-security

Purpose

External LAN security

Responsible Role

Network Administrator

Controlled Objects

comm:DateTime, comm:NtpProvider

ext-san-config

Purpose

External SAN configuration

Responsible Role

Storage Administrator

Controlled Objects

fabric:FcSan, fabric:FcSanEp, fabric:FcSanPc, fabric:FcSanPcEp, fabric:FcVsanPortEp, fabric:SanPinGroup, fabric:SanPinTarget, fabric:Vsan, fcpool:Format, vnic:FcOEIf

ext-san-policy

Purpose

External SAN policy

Responsible Role

Storage Administrator

Controlled Objects

fabric:FcSan, fabric:FcSanEp, fabric:FcSanPc, fabric:FcSanPcEp, fabric:FcVsanPortEp, fabric:SanPinGroup, fabric:SanPinTarget, fabric:Vsan, fcpool:Format, vnic:FcOEIf

ext-san-qos

Purpose

External SAN QoS

Responsible Role

Storage Administrator

Controlled Objects

qosclass:Definition, qosclass:EthBE, qosclass:EthClassified, qosclass:Fc

ext-san-security

Purpose

External SAN security

Responsible Role

Storage Administrator

Controlled Objects

There are no objects assigned to this privilege.

fault

Purpose

Alarms and alarm policies

Responsible Role

Operations

Controlled Objects

callhome:Policy, event:EpCtrl, event:Log, fault:Holder, fault:Inst, fault:Policy

ls-config

Purpose

Service profile configuration

Responsible Role

Server Profile Administrator

Controlled Objects

bios:VFeat, bios:VfConsoleRedirection, bios:VfEnhancedIntelSpeedStepTech, bios:VfFrontPanelLockout, bios:VfIntelHyperThreadingTech, bios:VfIntelTurboBoostTech, bios:VfIntelVTForDirectedIO, bios:VfIntelVirtualizationTechnology, bios:VfLvDIMMSupport, bios:VfMirroringMode, bios:VfNUMAOptimized, bios:VfProcessorC3Report, bios:VfProcessorC6Report, bios:VfQuietBoot, bios:VfResumeOnACPowerLoss, bios:VfSelectMemoryRASConfiguration, bios:VProfile, extvmm:Ep, extvmm:KeyRing, extvmm:KeyStore, extvmm:MasterExtKey, extvmm:Provider, extvmm:SwitchDelTask, ls:ComputeBinding, ls:Binding, ls:Requirement, ls:Power, ls:Server, ls:Tie, lsboot:Def, lsboot:Lan, lsboot:LanImagePath, lsboot:LocalStorage, lsboot:SanImage, lsboot:SanImagePath, lsboot:Storage, lsboot:VirtualMedia, org:Org, power:Group, power:Regulation, power:Rule, sol:Config,

storage:LocalDiskConfigDef, storage:LocalDiskPartition, vm:Cont, vm:DirCont, vm:DC, vm:DCOrg, vm:Org, vm:Switch, vm:DC, vm:DCOrg, vm:Org, vm:Switch, vm:VnicProfCl, vnic:BootTarget, vnic:DynamicCon, vnic:Ether, vnic:EtherIf, vnic:Fc, vnic:FcIf, vnic:FcOEIf, vnic:IPv4Dhcp, vnic:IPv4Dns, vnic:IPv4If, vnic:IPv4StaticRoute, vnic:IpV4PooledAddr, vnic:IpV4StaticAddr, vnic:Ipc, vnic:IpcIf, vnic:Scsi, vnic:ScsiIf

ls-config-policy

Purpose

Service profile configuration policy

Responsible Role

Server Profile Administrator

Controlled Objects

adaptor:EthCompQueueProfile, adaptor:EthFailoverProfile, adaptor:EthInterruptProfile, adaptor:EthOffloadProfile, adaptor:EthRecvQueueProfile, adaptor:EthWorkQueueProfile, adaptor:ExtIpV6RssHashProfile, adaptor:FcCdbWorkQueueProfile, adaptor:FcErrorRecoveryProfile, adaptor:FcInterruptProfile, adaptor:FcPortFLogiProfile, adaptor:FcPortPLogiProfile, adaptor:FcPortProfile, adaptor:FcRecvQueueProfile, adaptor:FcWorkQueueProfile, adaptor:HostEthIfProfile, adaptor:HostFcIfProfile, adaptor:IpV4RssHashProfile, adaptor:IpV6RssHashProfile, adaptor:RssProfile, extvmm:Ep, extvmm:KeyRing, extvmm:KeyStore, extvmm:MasterExtKey, extvmm:Provider, extvmm:SwitchDelTask, firmware:ComputeHostPack, firmware:ComputeMgmtPack, ls:AgentPolicy, ls:ComputeBinding, ls:Binding, ls:Requirement, ls:Tier, lsboot:Def, lsboot:Lan, lsboot:LanImagePath, lsboot:LocalStorage, lsboot:Policy, lsboot:SanImage, lsboot:SanImagePath, lsboot:Storage, lsboot:VirtualMedia, org:Org, sol:Config, sol:Policy, storage:LocalDiskConfigDef, storage:LocalDiskConfigPolicy, storage:LocalDiskPartition, vm:Cont, vm:DirCont, vm:DC, vm:DCOrg, vm:Org, vm:Switch, vm:DC, vm:DCOrg, vm:Org, vm:Switch, vm:VnicProfCl

ls-ext-access

Purpose

Service profile end point access

Responsible Role

Server Profile Administrator

This privilege is not used.

ls-network

Purpose

Service profile network

Responsible Role

Network Administrator

Controlled Objects

dpsec:Mac, extvmm:Provider, extvmm:SwitchDelTask, fabric:DceSwSrvEp, fabric:VCon, fabric:VConProfile, flowctrl:Definition, flowctrl:Item, macpool:Format, nwctrl:Definition, qos:Definition, epqos:Definition, epqos:DefinitionDelTask, qosclass:Definition, qos:Item, epqos:Item, epqos:Egress, qosclass:Item, qosclass:Eth, qosclass:EthBE, qosclass:EthClassified, qosclass:Fc, vm:Cont, vm:DirCont, vm:DC, vm:DCOrg, vm:Org, vm:Switch, vm:DC, vm:DCOrg, vm:Org, vm:Switch, vm:VnicProfCl, vnic:DefBeh, vnic:DynamicCon, vnic:DynamicConPolicy, vnic:DynamicIdUniverse, vnic:Ether, vnic:EtherIf, vnic:IPv4Dhcp, vnic:IPv4Dns, vnic:IPv4If, vnic:IPv4StaticRoute, vnic:IpV4PooledAddr, vnic:IpV4StaticAddr, vnic:Ipc, vnic:IpcIf, vnic:LanConnTempl, vnic:Profile, vnic:ProfileSet

ls-network-policy

Purpose

Service profile network policy

Responsible Role

Network Administrator

Controlled Objects

dpsec:Mac, fabric:DceSrv, fabric:DceSwSrv, fabric:DceSwSrvEp, fabric:EthDiag, fabric:FcDiag, fabric:VCon, fabric:VConProfile, flowctrl:Definition, flowctrl:Item, ippool:Block, ippool:Pool, macpool:Block, macpool:Format, macpool:Pool, nwctrl:Definition, qos:Definition, epqos:Definition, epqos:DefinitionDelTask, qosclass:Definition, qos:Item, epqos:Item, epqos:Egress, qosclass:Item, qosclass:Eth, qosclass:EthBE, qosclass:EthClassified, qosclass:Fc, uuidpool:Block, vnic:DynamicCon, vnic:DynamicConPolicy, vnic:DynamicIdUniverse, vnic:LanConnTempl, vnic:Profile, vnic:ProfileSet

ls-power

Purpose

Service profile power management

Responsible Role

Facility Manager

ls-qos

Purpose

Service profile

Responsible Role

QoS Network Administrator

This privilege is not used.

ls-qos-policy

Purpose

Service profile QoS policy

Responsible Role

Network Administrator

Controlled Objects

flowctrl:Definition, flowctrl:Item, qos:Definition, epqos:Definition, epqos:DefinitionDelTask, qosclass:Definition, qos:Item, epqos:Item, epqos:Egress, qosclass:Item, qosclass:Eth, qosclass:EthBE, qosclass:EthClassified, qosclass:Fc

ls-security

Purpose

Service profile security

Responsible Role

Server Security Administrator

Controlled Objects

aaa:EpAuthProfile, aaa:EpUser

ls-security-policy

Purpose

Service profile security policy

Responsible Role

Server Security Administrator

Controlled Objects

aaa:EpAuthProfile, aaa:EpUser

ls-server

Purpose

Service profile server management

Responsible Role

Server Security Administrator

Controlled Objects

bios:VFeat, bios:VfConsoleRedirection, bios:VfEnhancedIntelSpeedStepTech, bios:VfFrontPanelLockout, bios:VfIntelHyperThreadingTech, bios:VfIntelTurboBoostTech, bios:VfIntelVTForDirectedIO, bios:VfIntelVirtualizationTechnology, bios:VfLvDIMMSupport, bios:VfMirroringMode, bios:VfNUMAOptimized, bios:VfProcessorC3Report, bios:VfProcessorC6Report, bios:VfQuietBoot, bios:VfResumeOnACPowerLoss, bios:VfSelectMemoryRASConfiguration, bios:VProfile, ls:ComputeBinding, ls:Binding, ls:Requirement, ls:Power, ls:Server, ls:Tier, lsboot:Def, lsboot:Lan, lsboot:LanImagePath, lsboot:LocalStorage, lsboot:SanImage, lsboot:SanImagePath, lsboot:Storage, lsboot:VirtualMedia, power:Group, power:Regulation, power:Rule, sol:Config, storage:LocalDiskConfigDef, storage:LocalDiskPartition, vnic:BootTarget, vnic:DefBeh, vnic:DynamicCon, vnic:Ether, vnic:EtherIf, vnic:Fc, vnic:FcIf, vnic:FcNode, vnic:FcOEI, vnic:IPv4Dhcp, vnic:IPv4Dns, vnic:IPv4If, vnic:IPv4StaticRoute, vnic:IpV4PooledAddr, vnic:IpV4StaticAddr, vnic:Ipc, vnic:IpcIf, vnic:Scsi, vnic:ScsiIf

ls-server-oper

Purpose

Service profile consumer role

This privilege controls these operations on the service profile:

- Launch KVM
- Boot Server
- Shutdown Server
- Reset

Responsible Role

Server Profile Administrator

ls-server-policy

Purpose

Service profile pool policy

Responsible Role

Server Security Administrator

Controlled Objects

adaptor:EthCompQueueProfile, adaptor:EthFailoverProfile, adaptor:EthInterruptProfile, adaptor:EthOffloadProfile, adaptor:EthRecvQueueProfile, adaptor:EthWorkQueueProfile, adaptor:ExtIpV6RssHashProfile, adaptor:FcCdbWorkQueueProfile, adaptor:FcErrorRecoveryProfile, adaptor:FcInterruptProfile, adaptor:FcPortFLogiProfile, adaptor:FcPortPLogiProfile, adaptor:FcPortProfile, adaptor:FcRecvQueueProfile, adaptor:FcWorkQueueProfile, adaptor:HostEthIfProfile, adaptor:HostFcIfProfile, adaptor:IpV4RssHashProfile, adaptor:IpV6RssHashProfile, adaptor:RssProfile, bios:VFeat, bios:VfConsoleRedirection, bios:VfEnhancedIntelSpeedStepTech, bios:VfFrontPanelLockout, bios:VfIntelHyperThreadingTech, bios:VfIntelTurboBoostTech, ios:VfIntelVTForDirectedIO, bios:VfIntelVirtualizationTechnology, bios:VfLvDIMMSupport, bios:VfMirroringMode, bios:VfNUMAOptimized, bios:VfProcessorC3Report, bios:VfProcessorC6Report, bios:VfQuietBoot, bios:VfResumeOnACPowerLoss, bios:VfSelectMemoryRASConfiguration, bios:VProfile, fabric:VCon, fabric:VConProfile, firmware:ComputeHostPack, firmware:ComputeMgmtPack, ls:AgentPolicy, ls:ComputeBinding, ls:Binding, ls:Requirement, ls:Power, ls:Tier, lsboot:Policy, power:Group, power:Regulation, power:Rule

ls-storage

Purpose

Service profile storage

Responsible Role

Storage Administrator

Controlled Objects

fcpool:Format, lsboot:Def, lsboot:Lan, lsboot:LanImagePath, lsboot:LocalStorage, lsboot:SanImage, lsboot:SanImagePath, lsboot:Storage, lsboot:VirtualMedia, storage:LocalDiskConfigDef, storage:LocalDiskConfigPolicy, storage:LocalDiskPartition, uuidpool:Format, vnic:BootTarget, vnic:DefBeh, vnic:Fc, vnic:FcIf, vnic:FcNode, vnic:FcOEIf, vnic:SanConnTempl, vnic:Scsi, vnic:ScsiIf

ls-storage-policy

Purpose

Service profile storage policy

Responsible Role

Storage Administrator

Controlled Objects

fabric:VCon, fabric:VConProfile, fcpool:Block, fcpool:BootTarget, fcpool:Format, fcpool:Initiator, fcpool:Initiators, lsboot:Def, lsboot:Lan, lsboot:LanImagePath, lsboot:LocalStorage, lsboot:SanImage, lsboot:SanImagePath, lsboot:Storage, lsboot:VirtualMedia, storage:LocalDiskConfigDefstorage:LocalDiskConfigPolicy, storage:LocalDiskPartition, uuidpool:Format, vnic:SanConnTempl

operations

Purpose

Logs and Smart Call Home

Responsible Role

Operations

Controlled Objects

aaa:Log, callhome:Dest, callhome:Ep, callhome:PeriodicSystemInventory, callhome:Profile, callhome:Smtp, callhome:Source, callhome:TestAlert, comm:DateTime, comm:NtpProvider, comm:Syslog, comm:SyslogClient, comm:SyslogConsole, comm:SyslogFile, comm:SyslogMonitor, condition:Log, aaa:Log, event:Log, event:EpCtrl, event:Log, fault:Inst, stats:CollectionPolicy, stats:Curr, adaptor:EthPortBySizeLargeStats, adaptor:EthPortBySizeSmallStats, adaptor:EthPortErrStats, adaptor:EthPortMcastStats, adaptor:EthPortOutsizedStats, adaptor:EthPortStats, adaptor:EtherIfStats, adaptor:FcIfEventStats, adaptor:FcIfFC4Stats, adaptor:FcIfFrameStats, adaptor:FcPortStats, adaptor:MenloBaseErrorStats, adaptor:MenloDcePortStats, adaptor:MenloEthErrorStats, adaptor:MenloEthStats, adaptor:MenloFcErrorStats, adaptor:MenloFcStats, adaptor:MenloHostPortStats, adaptor:MenloMcpuErrorStats, adaptor:MenloMcpuStats, adaptor:MenloNetEgStats, adaptor:MenloNetInStats, adaptor:MenloQErrorStats, adaptor:MenloQStats, adaptor:VnicStats, compute:IOHubEnvStats, compute:MbPowerStats, compute:MbTempStats, compute:PCIeFatalCompletionStats, compute:PCIeFatalProtocolStats, compute:PCIeFatalReceiveStats, compute:PCIeFatalStats, equipment:ChassisStats, equipment:FanModuleStats, equipment:FanStats, equipment:IOCardStats, equipment:PsuInputStats, equipment:PsuStats, ether:ErrStats, ether:LossStats, ether:PauseStats, ether:RxStats, ether:TxStats, fc:ErrStats, fc:Stats, memory:ArrayEnvStats, memory:BufferUnitEnvStats, memory:ErrorStats, memory:Runtime, memory:UnitEnvStats, processor:EnvStats, processor:ErrorStats, processor:Runtime, sw:EnvStats, sw:SystemStats, stats:Holder, stats:Thr32Definition, stats:Thr32Value, stats:Thr64Definition, stats:Thr64Value, stats:ThrFloatDefinition, stats:ThrFloatValue, stats:ThresholdClass, stats:ThresholdDefinition, stats:Thr32Definition, stats:Thr64Definition, stats:ThrFloatDefinition, stats:ThresholdPolicy, stats:ThresholdValue, stats:Thr32Value, stats:Thr64Value, stats:ThrFloatValue, sysdebug:AutoCoreFileExportTarget, sysdebug:BackupBehavior, sysdebug:Core, sysdebug:CoreFileExportTarget, sysdebug:AutoCoreFileExportTarget, ysdebug:ManualCoreFileExportTarget), sysdebug:CoreFileRepository, sysdebug:LogControlDestinationFile, ysdebug:LogControlDestinationSyslog, sysdebug:LogControlDomain, sysdebug:LogControlEp, sysdebug:LogControlModule, sysdebug:MEpLog, sysdebug:MEpLogPolicy, sysdebug:ManualCoreFileExportTarget, sysfile:Mutation

pn-equipment

Purpose

Server hardware management

Responsible Role

Server Equipment Administrator

Controlled Objects

adaptor:ExtIf, adaptor:ExtEthIf, adaptor:HostIf, adaptor:HostEthIf, adaptor:HostFcIf, compute:Blade, compute:PsuPolicy, diag:SrvCtrl, equipment:Chassis, equipment:Led, equipment:IndicatorLed, equipment:LocatorLed, fabric:ComputeSlotEp, fabric:SwChPhEp

pn-maintenance

Purpose

Server maintenance

Responsible Role

Server Equipment Administrator

Controlled Objects

adaptor:ExtIf, adaptor:ExtEthIf, adaptor:HostIf, adaptor:HostEthIf, adaptor:HostFcIf, compute:Blade, diag:SrvCtrl, equipment:Chassis, equipment:Led, equipment:IndicatorLed, equipment:LocatorLed, fabric:ComputeSlotEp, fabric:SwChPhEp

pn-policy

Purpose

Server policy

Responsible Role

Server Equipment Administrator

Controlled Objects

adaptor:CapQual, adaptor:Qual, bios:VFeat, bios:VfConsoleRedirection, bios:VfEnhancedIntelSpeedStepTech, bios:VfFrontPanelLockout, bios:VfIntelHyperThreadingTech, bios:VfIntelTurboBoostTech, bios:VfIntelVTForDirectedIO, bios:VfIntelVirtualizationTechnology, bios:VfLvDIMMSupport, bios:VfMirroringMode, bios:VfNUMAOptimized, bios:VfProcessorC3Report, bios:VfProcessorC6Report, bios:VfQuietBoot, bios:VfResumeOnACPowerLoss, bios:VfSelectMemoryRASConfiguration, bios:VProfile,

compute:AutoconfigPolicy, compute:Blade, compute:BladeDiscPolicy, compute:BladeInheritPolicy, compute:ChassisDiscPolicy, compute:ChassisQual, compute:DiscPolicy, compute:BladeDiscPolicy, compute:ChassisDiscPolicy, compute:PhysicalQual, compute:Pool, compute:PooledPhysical, compute:PooledSlot, compute:PooledSlot, compute:PoolingPolicy, compute:PsuPolicy, compute:Qual, compute:QualItem, adaptor:CapDef, adaptor:CapQual, adaptor:CapSpec, adaptor:Qual, compute:BladePosQual, compute:ChassisQual, compute:SlotQual, compute:PhysicalQual, memory:Qual, processor:Qual, storage:Qual, compute:ScrubPolicy, compute:SlotQual, diag:BladeTest, diag:NetworkTest, diag:RunPolicy, equipment:Chassis, equipment:Led, equipment:IndicatorLed, equipment:LocatorLed, extvmm:Ep, extvmm:KeyRing, extvmm:KeyStore, extvmm:MasterExtKey, extvmm:Provider, extvmm:SwitchDelTask, fabric:ComputeSlotEp, fabric:SwChPhEp, memory:Qual, org:Org, processor:Qual, storage:Qual, uuidpool:Pool, vm:Cont, vm:DirCont, vm:DC, vm:DCOrg, vm:Org, vm:Switch, vm:DC, vm:DCOrg, vm:LifeCyclePolicy, vm:Org, vm:Switch, vm:VnicProfCl

pn-security

Purpose

Server security

Responsible Role

Server Security Administrator

Controlled Objects

mgmt:IntAuthPolicy

pod-config

Purpose

Pod configuration

Responsible Role

Network Administrator

This privilege is not used.

pod-policy

Purpose

Pod policy

Responsible Role

Network Administrator

This privilege is not used.

pod-qos

Purpose

Pod QoS

Responsible Role

Network Administrator

This privilege is not used.

pod-security

Purpose

Pod security

Responsible Role

Network Administrator

This privilege is not used.

power-mgmt

Purpose

Data center power management

This role provides read and write access for power capacity management including power group configurations and other power-related policies.

Responsible Role

Facility Manager

read-only

Purpose

Read-only access

Responsible Role

This is not a selectable privilege. All roles have read-only access to all objects. Roles that have read-write privileges on some objects also have read-only access to all other objects.

INDEX

Q

R

S

U

W