



## **Cisco UCS Manager XML API Programmer's Guide**

April 21, 2011

### **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

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at [www.cisco.com/go/trademarks](http://www.cisco.com/go/trademarks). Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1005R)

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

*Cisco UCS Manager XML API Programmer's Guide*  
© 2011 Cisco Systems, Inc. All rights reserved.



# CONTENTS

## **Preface** xi

- Audience xi
- Document Organization xi
- Related Documentation xii
- Documentation Feedback xii

---

## **CHAPTER 1**

### **Cisco UCS Manager XML API** 1-1

- About Cisco UCS Manager XML API 1-1
- Unified Computing System Overview 1-2
- Cisco UCS Management Information Model 1-2
- Cisco UCS XML API Sample Flow 1-3
- Object Naming 1-5
  - Distinguished Name 1-5
  - Relative Name 1-5
- API Method Categories 1-5
  - Authentication Methods 1-6
  - Query Methods 1-6
    - Query Filters 1-7
  - Configuration Methods 1-8
  - Event Subscription Methods 1-9
  - Capturing XML Interchange Between the GUI and the Cisco UCS 1-9
- Success or Failure Response 1-9
  - Successful Response 1-10
  - Failed Requests 1-11
  - Empty Results 1-11

---

## **CHAPTER 2**

### **Using the Cisco UCS XML API Methods** 2-1

- Authentication Methods 2-1
  - Login 2-1
  - Refreshing the Session 2-2
  - Logging Out of the Session 2-2
  - Unsuccessful Responses 2-3
- Query Methods 2-3

- Using configFindDnsByClassId 2-4
- Using configResolveChildren 2-4
- Using configResolveClass 2-4
- Using configResolveClasses 2-5
- Using configResolveDn 2-5
- Using configResolveDns 2-5
- Using configResolveParent 2-5
- Using configScope 2-6
- Querying the MAC Pool 2-6
- Using Query Methods for Statistics 2-7
- Querying Faults 2-8
- Using Filters 2-9
  - Simple Filters 2-9
  - Property Filters 2-9
    - Equality Filter 2-10
    - Not Equal Filter 2-10
    - Greater Than Filter 2-10
    - Greater Than or Equal to Filter 2-10
    - Less Than Filter 2-11
    - Less Than or Equal to Filter 2-11
    - Wildcard Filter 2-11
    - Any Bits Filter 2-11
    - All Bits Filter 2-12
  - Composite Filters 2-12
    - AND Filter 2-12
    - OR Filter 2-13
    - Between Filter 2-13
    - AND, OR, NOT Composite Filter 2-14
  - Modifier Filter 2-14
    - NOT Filter 2-14

**CHAPTER 3**

**Cisco UCS XML API Method Descriptions 3-1**

- API Method Descriptions 3-1
  - aaaChangeSelfPassword 3-1
    - Request Syntax 3-2
    - Response Syntax 3-2
    - Examples 3-3
  - aaaCheckComputeAuthToken 3-3
    - Request Syntax 3-3

Response Syntax	3-3
Examples	3-4
aaaCheckComputeExtAccess	3-5
Request Syntax	3-5
Response Syntax	3-5
Examples	3-5
aaaGetNComputeAuthTokenByDn	3-6
Request Syntax	3-6
Response Syntax	3-6
Examples	3-7
aaaKeepAlive	3-7
Request Syntax	3-7
Response Syntax	3-7
Examples	3-7
aaaLogin	3-8
Request Syntax	3-8
Response Syntax	3-8
Examples	3-9
aaaLogout	3-9
Request Syntax	3-10
Response Syntax	3-10
Examples	3-10
aaaRefresh	3-10
Request Syntax	3-10
Response Syntax	3-11
Examples	3-12
aaaTokenLogin	3-12
Request Syntax	3-12
Response Syntax	3-13
Examples	3-14
aaaTokenRefresh	3-14
Request Syntax	3-14
Response Syntax	3-14
Examples	3-15
configCheckConformance	3-16
Request Syntax	3-16
Response Syntax	3-16
Examples	3-16
configCheckFirmwareUpdatable	3-17
Request Syntax	3-17

Response Syntax	3-17
Examples	3-18
configConfFiltered	3-18
Request Syntax	3-18
Response Syntax	3-19
Examples	3-19
configConfMo	3-20
Request Syntax	3-20
Response Syntax	3-20
Examples	3-20
configConfMoGroup	3-21
Request Syntax	3-21
Response Syntax	3-22
Examples	3-22
configConfMos	3-23
Request Syntax	3-23
Response Syntax	3-23
Examples	3-24
configEstimateImpact	3-25
Request Syntax	3-25
Response Syntax	3-26
Examples	3-26
configFindDependencies	3-28
Request Syntax	3-28
Response Syntax	3-28
Examples	3-29
configFindDnsByClassId	3-30
Request Syntax	3-30
Response Syntax	3-30
Examples	3-31
configMoChangeEvent	3-31
Request Syntax	3-31
Response Syntax	3-31
Examples	3-32
configResolveChildren	3-33
Request Syntax	3-33
Response Syntax	3-33
Examples	3-33
configResolveClass	3-34
Request Syntax	3-34

Response Syntax	3-35
Examples	3-35
configResolveClasses	3-35
Request Syntax	3-36
Response Syntax	3-36
Examples	3-36
configResolveDn	3-37
Request Syntax	3-37
Response Syntax	3-37
Examples	3-38
configResolveDns	3-38
Request Syntax	3-39
Response Syntax	3-39
Examples	3-39
configResolveParent	3-40
Request Syntax	3-40
Response Syntax	3-40
Examples	3-41
configScope	3-42
Request Syntax	3-42
Response Syntax	3-43
Examples	3-43
eventSendHeartbeat	3-43
Request Syntax	3-43
Response Syntax	3-44
Examples	3-44
eventSubscribe	3-44
Request Syntax	3-44
Response Syntax	3-45
Examples	3-45
faultAckFault	3-45
Request Syntax	3-45
Response Syntax	3-45
Examples	3-46
faultAckFaults	3-46
Request Syntax	3-46
Response Syntax	3-46
Examples	3-46
faultResolveFault	3-47
Request Syntax	3-47

Response Syntax	3-47
Examples	3-47
IsClone	3-48
Request Syntax	3-48
Response Syntax	3-49
Examples	3-49
IsInstantiateNamedTemplate	3-51
Request Syntax	3-51
Response Syntax	3-52
Examples	3-52
IsInstantiateTemplate	3-53
Request Syntax	3-53
Response Syntax	3-54
Examples	3-54
IsInstantiateTemplate	3-55
Request Syntax	3-55
Response Syntax	3-56
Examples	3-56
IsResolveTemplates	3-57
Request Syntax	3-57
Response Syntax	3-58
Examples	3-59
IsTemplatise	3-59
Request Syntax	3-59
Response Syntax	3-60
Examples	3-60
orgResolveElements	3-61
Request Syntax	3-61
Response Syntax	3-62
Examples	3-62
poolResolveInScope	3-64
Request Syntax	3-64
Response Syntax	3-64
Examples	3-65
statsClearInterval	3-65
Request Syntax	3-65
Response Syntax	3-66
Examples	3-66
statsResolveThresholdPolicy	3-66
Request Syntax	3-66

Response Syntax 3-67

Examples 3-67

**CHAPTER 4**

**Cisco UCS XML Object-Access Privileges 4-1**

Privileges Summary Table 4-1

Privileges Description and Object List 4-2

aaa 4-2

admin 4-3

ext-lan-config 4-3

ext-lan-policy 4-3

ext-lan-qos 4-3

ext-lan-security 4-4

ext-san-config 4-4

ext-san-policy 4-4

ext-san-qos 4-4

ext-san-security 4-4

fault 4-5

ls-config 4-5

ls-config-policy 4-5

ls-ext-access 4-6

ls-network 4-6

ls-network-policy 4-6

ls-qos 4-6

ls-qos-policy 4-7

ls-security 4-7

ls-security-policy 4-7

ls-server 4-7

ls-server-policy 4-8

ls-storage 4-8

ls-storage-policy 4-8

operations 4-8

pn-equipment 4-9

pn-maintenance 4-9

pn-policy 4-10

pn-security 4-10

pod-config 4-10

pod-policy 4-11

pod-qos 4-11

pod-security 4-11

- read-only 4-11
- Power Management 4-11
  - power-mgmt 4-11
  - ls-server-oper 4-11
  - ls-power 4-12

---

**INDEX**



## Preface

---

This chapter includes the following:

- [Audience, page xi](#)
- [Document Organization, page xi](#)
- [Related Documentation, page xii](#)
- [Documentation Feedback, page xii](#)

## Audience

This guide is intended for software engineers with a background in programming and the use of APIs. Engineers should have knowledge of XML, data systems, networking protocols, and storage protocols.

## Document Organization

This XML API Reference Guide is organized into the following chapters:

- [Chapter 1, “Cisco UCS Manager XML API”](#)
- [Chapter 2, “Using the Cisco UCS XML API Methods”](#)
- [Chapter 3, “Cisco UCS XML API Method Descriptions”](#)
- [Chapter 4, “Cisco UCS XML Object-Access Privileges”](#)

## Related Documentation

For information on the Cisco UCS, visit the following product pages:

<http://www.cisco.com/en/US/products/ps10265/index.html>

<http://www.cisco.com/en/US/products/ps10281/index.html>

Documentation for Cisco UCS is available at the following URL:

[http://www.cisco.com/en/US/docs/unified\\_computing/ucs/overview/guide/UCS\\_roadmap.html](http://www.cisco.com/en/US/docs/unified_computing/ucs/overview/guide/UCS_roadmap.html)

**Note**

---

The *Cisco UCS Manager GUI Configuration Guide* and the *Cisco UCS Manager CLI Command Reference* provide an overview of the Unified Computing System and the UCS Manager. This is important background information for XML API software developers.

---

## 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](mailto:ucs-docfeedback@cisco.com). We appreciate your feedback.



# CHAPTER 1

## Cisco UCS Manager XML API

---

This chapter includes the following:

- [About Cisco UCS Manager XML API, page 1-1](#)
- [Unified Computing System Overview, page 1-2](#)
- [Cisco UCS Management Information Model, page 1-2](#)
- [Cisco UCS XML API Sample Flow, page 1-3](#)
- [Object Naming, page 1-5](#)
- [API Method Categories, page 1-5](#)
- [Success or Failure Response, page 1-9](#)

### About Cisco UCS Manager XML API

The Cisco UCS Manager XML API is a programmatic interface to the Cisco Unified Computing System. 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 of the Cisco UCS is stored in a hierarchical tree structure known as the management information tree, which is completely accessible through the XML API.

The API model is recursively driven and provides major functionality for application development. For example, changes can be made on a single object, an object subtree, or the entire object tree. With a single API call, changes can be made to a single attribute of an object, or to the entire Cisco UCS structure, including the configuration of chassis, blades, adapters, polices, and most other hardware and software 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 returns the configuration to its prior state, stops the API operation that listens for API requests, and sends a fault notification.

The API leverages an asynchronous operations model to improve scalability and performance. Slower API processes are nonblocking so that faster processes can proceed. A process receives a success message upon a valid request, and a complete message when the task is finished.

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

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.

## Unified Computing System Overview

A Cisco UCS unit can consist of up to two Cisco UCS fabric interconnects and a minimum of one Cisco chassis with one blade or rack-mounted server. Up to 40 chassis with a mixture of blade and rack-mounted servers can be connected and controlled by a single Cisco UCS Manager instance.

Cisco UCS Manager runs on the primary fabric interconnect, with failover capability to the subordinate fabric interconnect. In the event of a failover, the virtual IP address will connect to the subordinate fabric interconnect, making it the new primary fabric interconnect.

All XML requests to the Cisco UCS are asynchronous and terminate on the active Cisco UCS Manager. Cisco UCS Manager mediates all communication within the system; no direct user access to the Cisco UCS components is required.

Cisco UCS Manager is aware of the current configuration and performs automated device discovery whenever a new resource is installed. After a resource is detected, the Cisco UCS Manager adds it and its characteristics to the system inventory. Cisco UCS Manager can preconfigure the new resources if it is directed to do so by an administrator-defined policy.

## Cisco UCS 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 management information tree. 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 top (sys) 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 resources, such as fabric interconnects, chassis, blades, and rack-mounted servers.

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).

The information model is centrally stored and managed by the data management engine (DME), a user-level process running on the fabric interconnects. When a user initiates an administrative change to a Cisco UCS component (for example, applying a service profile to a server), the DME first applies that change to the information model, and then applies the change to the actual managed endpoint. This approach is called a model-driven framework.

Figure 1-1 is a branch diagram starting at sys from topRoot of the Cisco UCS management information tree. The diagram consists of five populated chassis with eight blades in each chassis, and each of the blades has one or more adapters. For simplicity, only chassis number five is expanded.

**Figure 1-1 Illustration of MIN Structure Showing Five Chassis**

Tree (topRoot):-----Distinguished Name:

```

|--sys----- (sys)
  |--chassis-1----- (sys/chassis-1)
  |--chassis-2----- (sys/chassis-2)
  |--chassis-3----- (sys/chassis-3)
  |--chassis-4----- (sys/chassis-4)
  |--chassis-5----- (sys/chassis-5)
    |--blade-1----- (sys/chassis-5/blade-1)
      |--adaptor-1-- (sys/chassis-5/blade-1/adaptor-1)
    |--blade-2----- (sys/chassis-5/blade-2)
      |--adaptor-1-- (sys/chassis-5/blade-2/adaptor-1)
      |--adaptor-2-- (sys/chassis-5/blade-2/adaptor-2)
    |--blade-3----- (sys/chassis-5/blade-3)
      |--adaptor-1-- (sys/chassis-5/blade-3/adaptor-1)
      |--adaptor-2-- (sys/chassis-5/blade-3/adaptor-2)
    |--blade-4----- (sys/chassis-5/blade-4)
      |--adaptor-1-- (sys/chassis-5/blade-4/adaptor-1)
    |--blade-5----- (sys/chassis-5/blade-5)
      |--adaptor-1-- (sys/chassis-5/blade-5/adaptor-1)
      |--adaptor-2-- (sys/chassis-5/blade-5/adaptor-2)
    |--blade-6----- (sys/chassis-5/blade-6)
      |--adaptor-1-- (sys/chassis-5/blade-6/adaptor-1)
    |--blade-7----- (sys/chassis-5/blade-7)
      |--adaptor-1-- (sys/chassis-5/blade-7/adaptor-1)
    |--blade-8----- (sys/chassis-5/blade-8)
      |--adaptor-1-- (sys/chassis-5/blade-8/adaptor-1)

```

## Cisco UCS 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.

Figure 1-2 shows how Cisco UCS Manager processes a boot server request.

Table 1-1 describes the steps involved in a boot server request.

Figure 1-2 Sample Flow of Boot Server Request

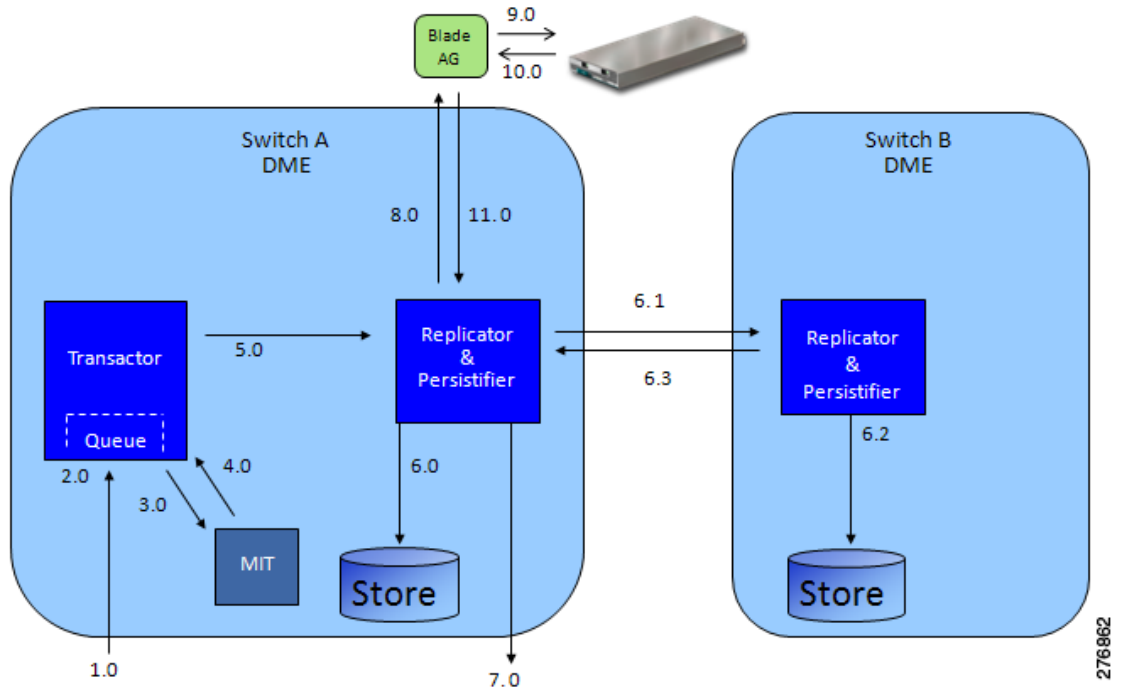


Table 1-1 Explanation of Boot Server Request

Step	Command/Process	Administrative Power State of MO (Server)	Operational Power State of MO (Server)
1	CMD request: boot server	Down	Down
2	Request queued	Down	Down
3	State change in management information tree	Up	Down
4	Transaction complete	Up	Down
5	Pass change information and boot request stimuli	Up	Down
6.0	Persistify (make persistent) the MO state change	Up	Down
6.1	Send state change information to peer DME	Up	Down
6.2	Persistify the MO state to peer's local store	Up	Down
6.3	Reply with success (replication and persistification)	Up	Down
7	CMD: response and external notification	Up	Down
8	Apply boot stimuli	Up	Down
9	Instruct BMC to power on server	Up	Down
10	Reply from BMC: server power on success	Up	Up
11	Reply, boot stimuli success, pass new power state information	Up	Up

# Object Naming

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

This section contains the following topics:

- [Distinguished Name, page 1-5](#)
- [Relative Name, page 1-5](#)

## 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.

This section contains the following topics:

- [Authentication Methods, page 1-6](#)
- [Query Methods, page 1-6](#)
- [Configuration Methods, page 1-8](#)
- [Event Subscription Methods, page 1-9](#)
- [Capturing XML Interchange Between the GUI and the Cisco UCS, page 1-9](#)

**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 Cisco UCS 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 current authentication cookie.

Authentication methods initiate and maintain an active session. A successful authentication must be performed before other API calls are allowed. API requests are cookie authenticated.

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.

Operations are performed using the HTTP post method (Cisco UCS supports both HTTP and HTTPS requests) over TCP. HTTP and HTTPS can be configured to use different port numbers, but TCP/80 (or TCP/443 for secure connections) is used by default. The HTTP envelope contains the XML configuration.

**Note**

The assumption is that the HTTP (or HTTPS) over TCP connection is handled by a scripting or programming language rather than a browser.

After a connection is established and authenticated, a cookie is returned in the response. The cookie is valid for 7200 seconds (120 minutes), and must be refreshed during the session period to prevent it from expiring. Each refresh operation creates a cookie valid for the default interval.

## Query Methods

Query methods obtain information on the current configuration state of an object. The following are query examples:

- `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"></>
```

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).

## Query Filters

The API 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.

This section contains the following topics:

- [Simple Filters, page 1-7](#)
- [Property Filters, page 1-7](#)
- [Composite Filters, page 1-8](#)
- [Modifier Filter, page 1-8](#)

### 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.

- NOT filter—Negates the result of a contained filter. Use this filter to obtain objects that do not match contained criteria.

This is the only modifier filter that is supported.

## 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 subtree (for example, a DN).
- 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

When an object is created, changed, or deleted because of a user- or system-initiated action, an event is generated. Applications get state change information by regular polling or event subscription. Because polling is resource-expensive, event subscription is the preferred method of notification.

On an event that has a subscription, Cisco UCS notifies the client and the type of event. Only actual change events are sent, not the object's unaffected attributes. This applies to all object changes in the system.

Every time an object is created, changed, or deleted because of a user-initiated or system-initiated action, an event is generated. Applications can get state change information by either of the following options:

- Event subscription (recommended)—Client application registers to receive event notifications.
- Polling—Expensive in terms of network resources. Use ONLY under very limited circumstances.

With subscription, when an event occurs, Cisco UCS sends a notice that contains only changed data. Notification can be sent for all object changes in the system. Use `eventSubscribe` to register for events, as shown the following example of subscriptions:

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

To receive notifications, open an HTTP or HTTPS session over TCP and keep the session open. On sending `eventSubscribe`, Cisco UCS starts sending all new events as they occur.

Each event has a unique event ID. 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. If the client detects a missing event ID, the client can use `eventSendEvent` to retrieve the missed event.

## Capturing XML Interchange Between the GUI and the Cisco UCS

Interchange is stored in a log file such as `C:\Documents and Settings\username\Application Data\Sun\Java\Deployment\log\ucsm`. 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

Cisco UCS responds almost immediately to any API request. The response indicates failure if the request is impossible to complete. A successful response indicates that the request is valid, but 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.

This section contains the following topics:

- [Successful Response, page 1-10](#)
- [Failed Requests, page 1-11](#)
- [Empty Results, page 1-11](#)

## Successful Response

When a request has executed successfully, Cisco UCS 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" />
```

The response includes the following information:

```
<configResolveDn dn="sys/chassis-1/blade-1"
  cookie="<real_cookie>"
  response="yes">
  <outConfig>
    <computeItem adminPower="policy"
      adminState="in-service"
      assignedToDn=" "
      association="none"
      availability="available"
      chassisId="1"
      checkPoint="discovered"
      connPath="A"
      connStatus="A"
      discovery="complete"
      dn="sys/chassis-1/blade-1"
      fltAggr="0"
      fsmDescr=" "
      fsmFlags=" "
      fsmPrev="DiscoverSuccess"
      fsmRmtInvErrCode="unspecified"
      fsmRmtInvErrDescr=" "
      fsmRmtInvRslt=" "
      fsmStageDescr=" "
      fsmStamp="2008-11-24T01:27:10"
      fsmStatus="nop"
      fsmTry="0"
      lc="discovered"
      managingInst="A"
      model="Gooding"
      name=" "
      numOfAdaptors="1"
      numOfCores="4"
      numOfEthHostIfs="2"
      numOfFcHostIfs="2"
      numOfThreads="0"
      operPower="off"
      operState="unassociated"
      operability="operable"
      originalUuid="1b4e28ba-2fa1-11d2-0101-b9a761bde3fb"
      presence="equipped"
      revision=" "
      serial="1-1"
      slotId="1"
      totalMemory="4096"
      uuid=" "
      vendor="Cisco" />
  </outConfig>
</configResolveDn>
```

## 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, Cisco UCS 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 blade-4711:

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





# CHAPTER 2

## Using the Cisco UCS XML API Methods

---

This chapter includes the following sections:

- [Authentication Methods, page 2-1](#)
- [Query Methods, page 2-3](#)
- [Using Query Methods for Statistics, page 2-7](#)
- [Querying Faults, page 2-8](#)
- [Using Filters, page 2-9](#)

### Authentication Methods

Authentication allows API interaction with the Cisco UCS. 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.

---

This section contains the following topics:

- [Login, page 2-1](#)
- [Refreshing the Session, page 2-2](#)
- [Logging Out of the Session, page 2-2](#)
- [Unsuccessful Responses, page 2-3](#)

### Login

The LogIn method uses a standard Telnet client to log in and be authenticated. A TCP session is established on port 80 and `aaaLogin` is called by posting a document that encapsulates the XML document in an HTTP message. The `/path` is always `/cisco`. To log in, establish a TCP connection as follows:

```
bash> telnet 172.21.60.72 80
POST /cisco HTTP/1.1
Host: 172.21.60.72
Content-Length: 52
```

Next call the `aaaLogin` method and provide a user name and password:

```
<aaaLogin
  inName="admin"
  inPassword="cisco@123" />
```


**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,
(6)       pod-policy,pod-qos,read-only"
(7)   outDomains="mgmt02-dummy"
(8)   outChannel="noencssl"
(9)   outEvtChannel="noencssl">
```

- (1) The method used to login.
- (2) Confirms this is a response.
- (3) This is the session cookie.
- (4) The recommended cookie refresh period. The default login session length is two hours.
- (5) The privileges assigned to the user account.
- (6) The `outDomains` value is `mgmt02-dummy`.
- (7) The `outChannel` value is `noencssl` (this session is not using encryption over SSL).
- (8) The `outEvtChannel` value is `noencssl` (any event subscriptions would not use encryption over SSL).
- (9) Closing tag.

## Refreshing the Session

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

```
<aaaRefresh
  inName="admin"
  inPassword="mypassword"
  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

Query methods obtain information (for example, hierarchy, state, and scope)

This section contains the following topics:

- [Using configFindDnsByClassId, page 2-4](#)
- [Using configResolveChildren, page 2-4](#)
- [Using configResolveClass, page 2-4](#)
- [Using configResolveClasses, page 2-5](#)
- [Using configResolveDn, page 2-5](#)
- [Using configResolveDns, page 2-5](#)
- [Using configResolveParent, page 2-5](#)
- [Using configScope, page 2-6](#)
- [Querying the MAC Pool, page 2-6](#)

## Using configFindDnsByClassId

When finding distinguished names of a specified class, note the following:

- This method retrieves the DNs of a specified class.
- *classId* specifies the object type to retrieve (required).
- Authentication cookie (from **aaaLogin** or **aaaRefresh**) is required.
- Enumerated values, classIds, and bit masks are displayed as strings.

See the example request/response in the [“configFindDnsByClassId” section on page 3-30](#).

## Using configResolveChildren

When resolving children of objects in the management information tree, 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 the [“configResolveChildren” section on page 3-33](#).

## 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 the [“configResolveClass” section on page 3-34](#).

## 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” section on page 3-35](#).

## 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” section on page 3-37](#).

## 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” section on page 3-38](#).

## 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](#)” section on page 3-40.

## 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](#)” section on page 3-42.

## 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>

```

## Using Query Methods for Statistics

Statistics are available on a wide range of objects. Querying all statistics at once is resource intensive. Instead, identify the type of statistic and the object on which it reports; for example, getting the *compCpuStats* object for *sys/chassis-1/blade-1/board/cpu-2*, query for all children of *sys/chassis-1/blade-1/board/cpu-2*. The request is as follows:

```

<configScope
  cookie="real_cookie"
  inHierarchical="false"
  dn="sys/chassis-1/blade-1/board/cpu-2"
  inClass="processorEnvStats" />

```

The system response is as follows:

```

<configScope
  dn="sys/chassis-1/blade-5/board/cpu-2"

```

```

cookie=<real_cookie>"
response="yes">
<outConfigs>
  <processorEnvStats
    dn="sys/chassis-1/blade-5/board/cpu-2/env-stats"
    inputVoltage="0.058200"
    inputVoltageAvg="0.062080"
    inputVoltageMax="0.077600"
    inputVoltageMin="0.058200"
    intervals="58982460"
    suspect="no"
    temperature="23.000000"
    temperatureAvg="23.000000"
    temperatureMax="23.000000"
    temperatureMin="23.000000"
    thresholded=""
    timeCollected="2009-09-23T12:40:55"
    update="327680"/>
  </outConfigs>
</configScope>

```

A query by the DN is more efficient:

```

<configResolveDn
  inHierarchical="false"
  cookie=<real_cookie>"
  dn="sys/chassis-1/blade-1/board/cpu-2/processorEnvStats">
</configResolveDn>

```

With the statistic object's DN, query for historical statistics on the children of the object using a hierarchical query. To get the statistic object and historical statistics, change *inHierarchical* to true:

```

<configResolveDn
  inHierarchical="true"
  cookie=<real_cookie>"
  dn="sys/chassis-1/blade-1/board/cpu-2/processorEnvStats">
</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

Use filters to create specific and targeted queries.

This section contains the following topics:

- [Simple Filters, page 2-9](#)
- [Property Filters, page 2-9](#)
- [Composite Filters, page 2-12](#)
- [Modifier Filter, page 2-14](#)

## 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

This section contains the following topics:

- [Equality Filter, page 2-10](#)
- [Not Equal Filter, page 2-10](#)
- [Greater Than Filter, page 2-10](#)
- [Greater Than or Equal to Filter, page 2-10](#)
- [Less Than Filter, page 2-11](#)
- [Less Than or Equal to Filter, page 2-11](#)
- [Wildcard Filter, page 2-11](#)
- [Any Bits Filter, page 2-11](#)
- [All Bits Filter, page 2-12](#)

## Equality Filter

The example shows a query for all associated servers. The equality 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 adaptor unit whose serial number begins with the prefix “QCII”:

```
<configResolveClass
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="adaptorUnit">
  <inFilter>
    <wcard class="adaptorUnit"
      property="serial"
      value="QCII*" />
  </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

This section includes the following topics:

- [AND Filter, page 2-12](#)
- [OR Filter, page 2-13](#)
- [Between Filter, page 2-13](#)
- [AND, OR, NOT Composite Filter, page 2-14](#)

## 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>

```

```

    </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>

```

## Modifier Filter

This section includes the following topic:

- [NOT Filter, page 2-14](#)

## NOT 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 XML API Method Descriptions

---

This chapter includes the following section:

- [API Method Descriptions, page 3-1](#)

### API Method Descriptions

These methods are also called from the GUI Console. This section provides API method descriptions, syntax (request and response) and a usage example. The API methods for the Cisco UCS are defined here.

- [aaaChangeSelfPassword](#)
- [aaaCheckComputeAuthToken](#)
- [aaaCheckComputeExtAccess](#)
- [aaaGetNComputeAuthTokenByDn](#)
- [aaaKeepAlive](#)
- [aaaLogin](#)
- [aaaLogout](#)
- [aaaRefresh](#)
- [aaaTokenLogin](#)
- [aaaTokenRefresh](#)
- [configCheckConformance](#)
- [configCheckFirmwareUpdatable](#)
- [configConfFiltered](#)
- [configConfMo](#)
- [configConfMoGroup](#)
- [configConfMos](#)
- [configEstimateImpact](#)
- [configFindDependencies](#)
- [configFindDnsByClassId](#)
- [configMoChangeEvent](#)
- [configResolveChildren](#)
- [configResolveClass](#)
- [configResolveClasses](#)
- [configResolveDn](#)
- [configResolveDns](#)
- [configResolveParent](#)
- [configScope](#)
- [eventSendHeartbeat](#)
- [eventSubscribe](#)
- [faultAckFault](#)
- [faultAckFaults](#)
- [faultResolveFault](#)
- [IsClone](#)
- [IsInstantiateNNamedTemplate](#)
- [IsInstantiateNTemplate](#)
- [IsInstantiateTemplate](#)
- [IsResolveTemplates](#)
- [IsTemplatis](#)
- [orgResolveElements](#)
- [poolResolveInScope](#)
- [statsClearInterval](#)
- [statsResolveThresholdPolicy](#)

### 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 and 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:complexType>
```

```

        <xs:attribute name="invocationResult" type="xs:string"/>
    </xs:complexType>

```

## Examples

### 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:complexType>

```

```

        </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>

```

## Examples

### Request

```

<aaaCheckComputeAuthToken
  cookie="<real_cookie>"
  inToken="04541875309302299687211"
  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: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>

```

### Examples

#### 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>

```

## Examples

### 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
  29538672765491844031,73106643969990280919791">
</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>
```

## Examples

### 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.

## 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>

```

## Examples

### Request

```

<aaaLogin
  inName="admin"
  inPassword="Nbv12345" />

```

### 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>
```

## Examples

### 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 7200 seconds that counts down when inactivity begins. If the 7200 seconds expire, Cisco UCS 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: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>

```

## Examples

### Request

```

<aaaRefresh
  cookie="<real_cookie>"
  inName="admin"
  inPassword="Nbv12345"
  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: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>

```

## Examples

### Request

```

<aaaTokenLogin
  inName="admin"
  inToken="80278502964410805791351" />

```

### 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>

```

## Examples

### 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>

```

## configCheckConformance

The configCheckConformance method checks if the given distributable (firmware package) can be used against the running Cisco UCS Manager version.

### Request Syntax

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

```
<xs:element name="configCheckConformance" type="configCheckConformance"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configCheckConformance" mixed="true">
    <xs:all>
      <xs:element name="outConfDns" type="dnSet" minOccurs="0"/>
      <xs:element name="outToResetDns" type="dnSet" minOccurs="0"/>
      <xs:element name="outNonConfDns" type="dnSet" minOccurs="0"/>
      <xs:element name="outInProgressDns" type="dnSet" minOccurs="0"/>
      <xs:element name="outNonUpgradableDns" 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:attribute name="dn" type="referenceObject"/>
  </xs:complexType>
```

### Examples

#### Request

```
<configCheckConformance
  dn="sys/fw-catalogue/distrib-ucs-k9-bundle-b-series.2.0.0.528.gbin"
  cookie="<real_cookie>"
/>
```

#### Response

```
<configCheckConformance
  dn="sys/fw-catalogue/distrib-ucs-k9-bundle-b-series.2.0.0.528.gbin"
  cookie="<real_cookie>"
  response="yes"
  errorCode="0"
  errorDescr=""
  <outConfDns>
    <dn value="sys/chassis-1/blade-5/mgmt/fw-system"/>
    <dn value="sys/chassis-1/blade-5/bios/fw-boot-loader"/>
    <dn value="sys/chassis-1/blade-3/boardController/mgmt/fw-system"/>
  </outConfDns>
  <outToResetDns>
  </outToResetDns>
```

```

<outNonConfDns>
  <dn value="sys/chassis-1/blade-1/mgmt/fw-system" />
  <dn value="sys/chassis-1/blade-3/mgmt/fw-system" />
  <dn value="sys/chassis-1/blade-1/bios/fw-boot-loader" />
  <dn value="sys/chassis-1/blade-3/bios/fw-boot-loader" />
  <dn value="sys/chassis-1/blade-3/adaptor-1/mgmt/fw-system" />
  <dn value="sys/chassis-1/blade-1/adaptor-2/mgmt/fw-system" />
  <dn value="sys/chassis-1/blade-1/adaptor-1/mgmt/fw-system" />
  <dn value="sys/chassis-1/blade-3/adaptor-2/mgmt/fw-system" />
  <dn value="sys/chassis-1/blade-5/adaptor-1/mgmt/fw-system" />
  <dn value="sys/chassis-1/blade-3/board/storage-SAS-1/fw-system" />
</outNonConfDns>
<outInProgressDns>
</outInProgressDns>
<outNonUpgradableDns>
</outNonUpgradableDns>
</configCheckConformance>

```

## configCheckFirmwareUpdatable

The configCheckFirmwareUpdatable method checks if firmware in certain components can be updated or activated. The method is triggered every time a user initiates an update or activate process.

For example, if a user tries to update the firmware version of an endpoint for which a firmware policy is specified as part of a service profile (either a host firmware pack or management firmware pack), the operation is disallowed. This method performs the validation.

### Request Syntax

```

<xs:element name="configCheckFirmwareUpdatable" type="configCheckFirmwareUpdatable"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configCheckFirmwareUpdatable" mixed="true">
    <xs:all>
      <xs:element name="inUpdatableDns" 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="configCheckFirmwareUpdatable" type="configCheckFirmwareUpdatable"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configCheckFirmwareUpdatable" mixed="true">
    <xs:all>
      <xs:element name="outPassDns" type="dnSet" minOccurs="0"/>
      <xs:element name="outFailDns" type="dnSet" minOccurs="0"/>
      <xs:element name="outInvalidDns" 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>

```

## Examples

### Request

```
<configCheckFirmwareUpdatable
  cookie="<real_cookie>">
  <inUpdatableDns>
    <dn value="sys/chassis-1/blade-5/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-5/adaptor-2/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-2/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-2/adaptor-1/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-1/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-3/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-3/adaptor-2/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-1/adaptor-1/mgmt/fw-updatable" />
  </inUpdatableDns>
</configCheckFirmwareUpdatable>
```

### Response

```
<configCheckFirmwareUpdatable
  cookie="<real_cookie>"
  response="yes">
  <outPassDns>
    <dn value="sys/chassis-1/blade-1/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-2/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-3/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-5/mgmt/fw-updatable" />
  </outPassDns>
  <outFailDns>
    <dn value="sys/chassis-1/blade-5/adaptor-2/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-2/adaptor-1/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-1/adaptor-1/mgmt/fw-updatable" />
    <dn value="sys/chassis-1/blade-3/adaptor-2/mgmt/fw-updatable" />
  </outFailDns>
  <outInvalidDns>
  </outInvalidDns>
</configCheckFirmwareUpdatable>
```

## configConfFiltered

The configConfFiltered method limits data and activity according to the configured policies.

### Request Syntax

```
<xs:element name="configConfFiltered" type="configConfFiltered"
  substitutionGroup="externalMethod"/>
  <xs:complexType name="configConfFiltered" mixed="true">
    <xs:all>
      <xs:element name="inFilter" type="filterFilter" minOccurs="0"/>
      <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:complexType>
```

```

        </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="configConfFiltered" type="configConfFiltered"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configConfFiltered" 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>

```

## Examples

### Request

```

<configConfFiltered
  cookie="<real_cookie>"
  inHierarchical="false"
  classId="orgTenant">
  <inFilter>
    <eq class="orgTenant" property="name" value="Cisco" />
  </inFilter>
  <inConfig>
    <orgDatacenter dn="org-HR" descr="HR (Human Resources- new Descr)"/>
  </inConfig>
</configConfFiltered>

```

### Response

```

<configConfFiltered
  cookie="<real_cookie>"
  commCookie="5/15/0/617"
  srcExtSys="10.193.33.206"
  destExtSys="10.193.33.206"
  srcSvc="sam_extXMLApi"
  destSvc="resource-mgr_dme"
  response="yes"
  classId="orgTenant">
  <outConfigs>
    <orgDatacenter
      descr="HR (Human Resources- new Descr) "
      dn="org-root/org-Cisco/org-HR"
      fltAggr="0"
      level="2"
      name="HR"
      status="modified" />
  </outConfigs>
</configConfFiltered>

```

## 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: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>
```

## Examples

### 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>

```

## configConfMoGroup

The configConfMoGroup method configures groups of managed objects based upon the configured policies.

### Request Syntax

```

<xs:element name="configConfMoGroup" type="configConfMoGroup"
  substitutionGroup="externalMethod"/>
  <xs:complexType name="configConfMoGroup" mixed="true">
    <xs:all>
      <xs:element name="inDns" type="dnSet" minOccurs="0"/>
      <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:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>

```

```

        <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="configConfMoGroup" type="configConfMoGroup"
substitutionGroup="externalMethod" />
  <xs:complexType name="configConfMoGroup" 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>

```

## Examples



### Note

The descr property of orgDataCenter (under org-root/org-Cisco and org-root/org-Soda) is modified. Because the descr property is not implicit, it can be modified. If implicit, the modification does not apply and a new orgDataCenter is created.

### Request

```

<configConfMoGroup
  cookie="<real_cookie">
  inHierarchical="false">
  <inDns>
    <dn value="org-root/org-Cisco" />
    <dn value="org-root/org-Soda" />
  </inDns>
  <inConfig>
    <orgDatacenter dn="org-HR" descr="HR (Human Resources)" />
  </inConfig>
</configConfMoGroup>

```

### Response

```

<configConfMoGroup
  cookie="<real_cookie">
  commCookie="5/15/0/600"
  srcExtSys="10.193.33.206"
  destExtSys="10.193.33.206"
  srcSvc="sam_extXMLApi"
  destSvc="resource-mgr_dme"
  response="yes">
  <outConfigs>
    <orgDatacenter
      descr="HR (Human Resources)"
      dn="org-root/org-Soda/org-HR"

```

```

        fltAggr="0"
        level="2"
        name="HR"
        status="modified"/>

    <orgDatacenter
        descr="HR (Human Resources)"
        dn="org-root/org-Cisco/org-HR"
        fltAggr="0"
        level="2"
        name="HR"
        status="modified"/>
</outConfigs>
</configConfMoGroup>

```

## 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: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>

```

## Examples

### 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>
```

## configEstimateImpact

The configEstimateImpact 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.

User can estimate the impact of a change set by passing the set to the method and inspecting the output parameters. Output parameters are a set of affected service profiles (before and after the changes) and the corresponding ack object for each service profile.

Ack objects contain the following information:

- Whether the changes are disruptive (for example, require reboot of the server associated to the service profile).
- Summary of changes.
- When changes are applied (immediately, after user ack, during scheduled occurrence of a maintenance window).
- Date and time at which such changes were made and by whom.

Cisco UCS returns the ack objects before and after the changes are applied. This information helps determine whether some changes were already pending on the service profile. This condition can occur when maintenance policies are used.

The parameters are defined as:

- configs—Set of changes to be evaluated (add, delete, or modify managed objects).
- affected—Affected service profiles after the changes have been applied (not hierarchical).
- oldAffected—Affected service profiles before applying changes (not hierarchical).
- ackables—Content of the ack object associated to the service profiles, after applying the changes.
- oldAckables—Content of the ack object associated to the service profiles, before applying the changes.

## Request Syntax

```
<xs:element name="configEstimateImpact" type="configEstimateImpact"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configEstimateImpact" 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="configEstimateImpact" type="configEstimateImpact"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configEstimateImpact" mixed="true">
    <xs:all>
      <xs:element name="outAckables" type="configMap" minOccurs="0">
        <xs:unique name="unique_map_key_6">
          <xs:selector xpath="pair"/>
          <xs:field xpath="@key"/>
        </xs:unique>
      </xs:element>
      <xs:element name="outOldAckables" type="configMap" minOccurs="0">
        <xs:unique name="unique_map_key_7">
          <xs:selector xpath="pair"/>
          <xs:field xpath="@key"/>
        </xs:unique>
      </xs:element>
      <xs:element name="outAffected" type="configMap" minOccurs="0">
        <xs:unique name="unique_map_key_8">
          <xs:selector xpath="pair"/>
          <xs:field xpath="@key"/>
        </xs:unique>
      </xs:element>
      <xs:element name="outOldAffected" type="configMap" minOccurs="0">
        <xs:unique name="unique_map_key_9">
          <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>
```

## Examples

### Request

```
<configEstimateImpact
  cookie="<real_cookie>"
  <inConfigs>
    <pair key="org-root/ls-template-3">
      <lsServer
        agentPolicyName=" "
        biosProfileName=" "
        bootPolicyName=" "
        descr=" "
        dn="org-root/ls-template-3"
        dynamicConPolicyName=" "
        extIPState="none"
        hostFwPolicyName=" "
        identPoolName="default"
        localDiskPolicyName=" "
        maintPolicyName=" "
        mgmtAccessPolicyName=" "
        mgmtFwPolicyName=" "
        name="template-3"
        powerPolicyName="default"
```

```

        scrubPolicyName=" "
        solPolicyName=" "
        srcTemplName=" "
        statsPolicyName="default"
        status="created"
        type="updating-template"
        usrLbl=" "
        uuid="derived"
        vconProfileName=" " />
    </pair>
</inConfigs>
</configEstimateImpact>

```

## Response

```

<configEstimateImpact
  cookie="<real_cookie>"
  response="yes"
  errorCode="0"
  errorDescr=" ">
  <outAckables>
  </outAckables>
  <outOldAckables>
  </outOldAckables>
  <outAffected>
    <pair key="org-root/ls-template-3">
      <lsServer
        agentPolicyName=" "
        assignState="unassigned"
        assocState="unassociated"
        biosProfileName=" "
        bootPolicyName=" "
        configQualifier=" "
        configState="not-applied"
        descr=" "
        dn="org-root/ls-template-3"
        dynamicConPolicyName=" "
        extIPState="none"
        fltAggr="0"
        hostFwPolicyName=" "
        identPoolName="default"
        intId="52359"
        localDiskPolicyName=" "
        maintPolicyName=" "
        mgmtAccessPolicyName=" "
        mgmtFwPolicyName=" "
        name="template-3"
        operBiosProfileName=" "
        operBootPolicyName="org-root/boot-policy-default"
        operDynamicConPolicyName=" "
        operHostFwPolicyName=" "
        operIdentPoolName="org-root/uuid-pool-default"
        operLocalDiskPolicyName="org-root/local-disk-config-default"
        operMaintPolicyName="org-root/maint-default"
        operMgmtAccessPolicyName=" "
        operMgmtFwPolicyName=" "
        operPowerPolicyName="org-root/power-policy-default"
        operScrubPolicyName="org-root/scrub-default"
        operSolPolicyName=" "
        operSrcTemplName=" "
        operState="unassociated"
        operStatsPolicyName="org-root/thr-policy-default"
        operVconProfileName=" "
        owner="management"

```

```

        pnDn=" "
        powerPolicyName="default"
        scrubPolicyName=" "
        solPolicyName=" "
        srcTemplName=" "
        statsPolicyName="default"
        status="created"
        type="updating-template"
        usrLbl=" "
        uuid="derived"
        uuidSuffix="0000-000000000000"
        vconProfileName=" " />
    </pair>
</outAffected>
<outOldAffected>
    <pair key="org-root/ls-template-3">
        <lsServer
            .
            ./>
        </pair>
    </outOldAffected>
</configEstimateImpact>

```

## 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>

```

## Examples

### 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"
    </lsServer>
  </outConfigs>
</configFindDependencies>

```

```

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>

```

## configFindDnsByClassId

The configFindDnsByClassId method finds distinguished names and returns them sorted by class ID.

### Request Syntax

```

<xs:element name="configFindDnsByClassId" type="configFindDnsByClassId"
substitutionGroup="externalMethod"/>
  <xs:complexType name="configFindDnsByClassId" 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:attribute name="classId" type="namingClassId"/>
  </xs:complexType>

```

### Response Syntax

```

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

```

```

<xs:all>
  <xs:element name="outDns" 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:attribute name="classId" type="namingClassId"/>
</xs:complexType>

```

## Examples

### Request

```

<configFindDnsByClassId
  classId="computeItem"
  cookie="<real_cookie>" />

```

### Response

```

<configFindDnsByClassId
  cookie="<real_cookie>"
  response="yes"
  classId="computeItem">
  <outDns>
    <dn value="sys/chassis-1/blade-7"/>
    <dn value="sys/chassis-1/blade-5"/>
    <dn value="sys/chassis-1/blade-3"/>
    <dn value="sys/chassis-1/blade-1"/>
  </outDns>
</configFindDnsByClassId>

```

## configMoChangeEvent

The configMoChangeEvent method provides event details from Cisco UCS 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 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:complexType>

```

```

<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>

```

## Examples

### 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: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>
```

### Examples

#### Request

```
<configResolveChildren
  cookie="<real_cookie>"
  classId="aaaUser"
  inDn="sys/user-ext"
  inHierarchical="false">
  <inFilter>
  </inFilter>
</configResolveChildren>
```

## Response

```

<configResolveChildren
  cookie="<real_cookie>"
  commCookie="11/15/0/2a59"
  srcExtSys="10.193.33.120"
  destExtSys="10.193.33.120"
  srcSvc="sam_extXMLApi"
  destSvc="mgmt-controller_dme"
  response="yes"
  classId="aaaUser">
<outConfigs>
  <aaaUser descr=" " dn="sys/user-ext/user-chambers"
    email=" " expiration="never" expires="no" firstName="John" intId="12716"
    lastName="Chambers" name="chambers" phone=" " priv="admin,read-only"
    pwdSet="yes"/>
  <aaaUser descr=" " dn="sys/user-ext/user-jackson" email=" " expiration="never"
    expires="no" firstName="Andrew" intId="12734" lastName="Jackson"
    name="jackson" phone=" "
    priv="fault,operations,policy,read-only,res-config,tenant" pwdSet="yes"/>
  <aaaUser descr=" " dn="sys/user-ext/user-admin" email=" " expiration="never"
    expires="no" firstName=" " intId="10052" lastName=" " name="admin" phone=" "
    priv="admin,read-only" pwdSet="yes"/>
  <aaaUser descr=" " dn="sys/user-ext/user-bama" email=" " expiration="never"
    expires="no" firstName="Rak" intId="12711" lastName="Bama" name="bama"
    phone=" " priv="fault,operations,policy,read-only,res-config,tenant"
    pwdSet="yes"/>
  <aaaUser descr=" " dn="sys/user-ext/user-fuld" email=" " expiration="never"
    expires="no" firstName="Richard" intId="12708" lastName="Fuld" name="fuld"
    phone=" " priv="read-only" pwdSet="yes"/>
  <aaaUser descr="testuser" dn="sys/user-ext/user-aaa" email=" "
    expiration="never" expires="no" firstName="a" intId="10620" lastName="aa"
    name="aaa" phone=" " priv="aaa,read-only" pwdSet="no"/>
  </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:complexType>

```

```

    </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>

```

## Examples

### Request

```

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

```

### Response

```

<configResolveClass
  cookie="<real_cookie>"
  commCookie="11/15/0/2a5b"
  srcExtSys="10.193.33.120"
  destExtSys="10.193.33.120"
  srcSvc="sam_extXMLApi"
  destSvc="mgmt-controller_dme"
  response="yes"
  classId="pkiEp">
  <outConfigs>
    <pkiEp descr=" "
      dn="sys/pki-ext"
      intId="10037"
      name=" "/>
  </outConfigs>
</configResolveClass>

```

## 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>
```

## Examples

### Request

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

### Response

(This is an abbreviated response.)

```
<configResolveClasses
  cookie="<real_cookie>"
  response="yes">
  <outConfigs>
    <computeItem
      adminPower="policy"
```

```

        adminState="in-service"
        dn="sys/chassis-1/blade-1"
        .
    ./>
<computeItem
    adminPower="policy"
    adminState="in-service"
    dn="sys/chassis-1/blade-3"
    .
    ./>
<computeItem
    adminPower="policy"
    adminState="in-service"
    dn="sys/chassis-1/blade-5"
    .
    ./>
<computeItem
    adminState="acknowledged"
    configState="ok"
    .
    ./>
</outConfigs>
</configResolveClasses>

```

## configResolveDn

For a specified DN, the configResolveDn method retrieves a single managed object.

### 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:complexType>

```

```

<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>

```

## Examples

### Request

```

<configResolveDn
  cookie="<real_cookie>"
  dn="vmmEp/vm-mgr-vcenter1" />

```

### Response

```

<configResolveDn dn="vmmEp/vm-mgr-vcenter1"
  cookie="<real_cookie>"
  commCookie="9/15/0/1c0d"
  srcExtSys="10.193.34.70"
  destExtSys="10.193.34.70"
  srcSvc="sam_extXMLApi"
  destSvc="vm-mgr_dme"
  response="yes">
  <outConfig>
    <vmManager
      adminState="enable"
      descr=" "
      dn="vmmEp/vm-mgr-vcenter1"
      fltAggr="0"
      fsmDescr="AG registration with
vCenter (FSM:sam:dme:VmManagerRegisterWithVCenter) "
      fsmPrev="RegisterWithVCenterRegistering"
      fsmProgr="13"
      fsmRmtInvErrCode="none"
      fsmRmtInvErrDescr=" "
      fsmRmtInvRslt=" "
      fsmStageDescr="AG registration with
vCenter (FSM-STAGE:sam:dme:VmManagerRegisterWithVCenter:Registering) "
      fsmStamp="2010-11-11T21:37:15.696"
      fsmStatus="RegisterWithVCenterRegistering"
      fsmTry="1"
      hostName="savbu-vpod-dev-31.cisco.com"
      intId="21959"
      name="vcenter1"
      operState="unknown"
      stateQual=" "
      type="vmware"
      version=" " />
    </outConfig>
  </configResolveDn>

```

## configResolveDns

For a list of DNSs, the configResolveDns method retrieves managed objects.

## 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: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>
```

## Examples

### Request

```
<configResolveDns
  cookie="<real_cookie>"
  inHierarchical="false">
  <inDns>
    <dn value="sys/chassis-1" />
    <dn value="sys/chassis-1/blade-1/board/cpu-1" />
    <dn value="sys/chassis-1/blade-1/board/t-stats" />
  </inDns>
</configResolveDns>
```

### Response

```
<configResolveDns
  cookie="<real_cookie>"
  response="yes">
  <outConfigs>
    <processorUnit
```

```

arch="Xeon"
cores="4"
dn="sys/chassis-1/blade-1/board/cpu-1"
id="1"
model="Intel(R) Xeon(R) CPU E5520 @ 2.27GHz"
operState="operable"
operability="operable"
perf="not-supported"
power="not-supported"
presence="equipped"
revision="0"
serial=""
socketDesignation="CPU1"
speed="2.266000"
stepping="5"
thermal="ok"
threads="8"
vendor="Intel(R) Corporation"
voltage="ok"/>
<equipmentChassis
.
.>
</outConfigs>
<outUnresolved>
  <dn value="sys/chassis-1/blade-1/board/t-stats"/>
</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>

```

## Examples

### Request

```

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

```

### Response

```

<configResolveParent
  dn="sys/chassis-1/blade-1/adaptor-1"
  cookie="<real_cookie>"
  response="yes">
  <outConfig>
    <computeItem
      adminPower="policy"
      adminState="in-service"
      assignedToDn=" "
      association="none"
      availability="available"
      availableMemory="10240"
      chassisId="1"
      checkPoint="discovered"
      connPath="A,B"
      connStatus="A,B"
      descr=" "
      diagnostics="complete"
      discovery="complete"
      dn="sys/chassis-1/blade-1"
      fltAggr="0"
      fsmDescr=" "
      fsmFlags=" "
      fsmPrev="DiscoverSuccess"
      fsmProgr="100"
      fsmRmtInvErrCode="none"
      fsmRmtInvErrDescr=" "
      fsmRmtInvRslt=" "
      fsmStageDescr=" "
      fsmStamp="2009-09-23T23:44:30"
      fsmStatus="nop"
      fsmTry="0"
      intId="768052"
      lc="discovered"
      lcTs="1969-12-31T16:00:00"
      managingInst="B"
      model="N20-B6620-1"
      name=" "
      numofAdaptors="1"
      numofCores="8"
    >
  </outConfig>
</configResolveParent>

```

```

        numOfCpus="2"
        numOfEthHostIfs="2"
        numOfFcHostIfs="0"
        numOfThreads="16"
        operPower="off"
        operQualifier=""
        operState="unassociated"
        operability="operable"
        originalUuid="e3516842-d0a4-11dd-baad-000bab01bfd6"
        presence="equipped"
        revision="0"
        serial="QCI12520024"
        slotId="1"
        totalMemory="10240"
        uuid="e3516842-d0a4-11dd-baad-000bab01bfd6"
        vendor="Cisco Systems Inc"/>
    </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: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: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>
```

## Examples

### Request

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

### Response

```
<configScope dn="org-root"
  cookie="<real_cookie>"
  commCookie="2/15/0/2a53"
  srcExtSys="10.193.33.120"
  destExtSys="10.193.33.120"
  srcSvc="sam_extXMLApi"
  destSvc="service-reg_dme"
  response="yes">
  <outConfigs>
    <orgOrgCaps dn="org-root/org-caps" org="512" tenant="64"/>
    <orgOrgCounts dn="org-root/org-counter" org="36" tenant="7"/>
  </outConfigs>
</configScope>
```

## eventSendHeartbeat

The eventSendHeartbeat method allows clients to retrieve any missed event. Each event has a unique event ID. These event IDs operate as counters and are included in all method responses.

Each time an event is generated, the event ID counter increases and the new event is assigned a new event ID. This enables the subscriber to track the events. If an event is missed by the client, the client can use the **eventSendEvent** method to retrieve the missed event.

## Request Syntax

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

```

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

```

## Response Syntax

```

<xs:element name="eventSendHeartbeat" type="eventSendHeartbeat"
  substitutionGroup="externalMethod"/>
  <xs:complexType name="eventSendHeartbeat" mixed="true">
    <xs:attribute name="outSystemTime" type="dateTime" />
    <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>

```

## Examples

### Request

When the client application subscribes to an event or events by using **eventSubscribe**, Cisco UCS sends **eventSendHeartbeat** periodically (default 120 seconds).

### Response

```

<eventSendHeartbeat
  cookie="<real_cookie>"
  commCookie=""
  srcExtSys="0.0.0.0"
  destExtSys="0.0.0.0"
  srcSvc=""
  destSvc=""
  response="yes"
  outSystemTime="2010-11-12T20:38:19.630">
</eventSendHeartbeat>

```

## eventSubscribe

The **eventSubscribe** method allows a client to subscribe to asynchronous events generated by Cisco UCS, 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. When an event occurs, Cisco UCS 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.

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

```

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

```

## 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>

```

## Examples

### Request

```

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

```

### Response

NO RESPONSE OR ACKNOWLEDGEMENT.

## faultAckFault

The faultAckFault method acknowledges a fault. The acknowledgement response marks the fault severity as cleared. Faults categorized as auto-cleared do not require acknowledgment.

## Request Syntax

```

<xs:element name="faultAckFault" type="faultAckFault" substitutionGroup="externalMethod"/>
  <xs:complexType name="faultAckFault" mixed="true">
    <xs:attribute name="inId" type="xs:unsignedLong"/>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>

```

## Response Syntax

```

<xs:element name="faultAckFault" type="faultAckFault" substitutionGroup="externalMethod"/>
  <xs:complexType name="faultAckFault" 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>

```

## Examples

### Request

```
<faultAckFault
  inHierarchical="false"
  cookie="<real_cookie>"
  inId="10120" />
```

### Response

```
<faultAckFault
  cookie="<real_cookie>"
  commCookie="5/15/0/6c"
  srcExtSys="10.193.33.214"
  destExtSys="10.193.33.214"
  srcSvc="sam_extXMLApi"
  destSvc="resource-mgr_dme"
  response="yes">
</faultAckFault>
```

## faultAckFaults

The faultAckFaults method acknowledges multiple faults. The acknowledgement response marks the fault severity as cleared. Faults categorized as auto-cleared do not require acknowledgment.

## Request Syntax

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

## Examples

### Request

```
<faultAckFaults
  cookie="<real_cookie>"
  <inIds>
    <id value="10656"/>
```

```

        <id value="10660"/>
    </inIds>
</faultAckFaults>

```

### Response

```

<faultAckFaults
  cookie="<real_cookie>"
  commCookie="11/15/0/505"
  srcExtSys="10.193.34.70"
  destExtSys="10.193.34.70"
  srcSvc="sam_extXMLApi"
  destSvc="mgmt-controller_dme"
  response="yes">
</faultAckFaults>

```

## faultResolveFault

The faultResolveFault method sends a response when a fault has been resolved.

### Request Syntax

```

<xs:element name="faultResolveFault" type="faultResolveFault"
  substitutionGroup="externalMethod"/>
  <xs:complexType name="faultResolveFault" mixed="true">
    <xs:attribute name="inId" type="xs:unsignedLong"/>
    <xs:attribute name="cookie" type="xs:string"/>
    <xs:attribute name="response" type="YesOrNo"/>
  </xs:complexType>

```

### Response Syntax

```

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

```

## Examples

### Request

```

<faultResolveFault
  inHierarchical="false"
  cookie="<real_cookie>"
  inId="10120" />

```

### Response

```

<faultResolveFault
  cookie="<real_cookie>"

```

```

commCookie="5/15/0/6a"
srcExtSys="10.193.33.214"
destExtSys="10.193.33.214"
srcSvc="sam_extXMLApi"
destSvc="resource-mgr_dme"
response="yes">
<outFault>
  <faultInst
    ack="yes"
    cause="empty-pool"
    changeSet=""
    code="F0135"
    created="2010-11-19T11:02:41.568"
    descr="Virtual Security Gateway pool default is empty"
    dn="org-root/fwpool-default/fault-F0135"
    highestSeverity="minor"
    id="10120"
    lastTransition="2010-11-19T11:02:41.568"
    lc=""
    occur="1"
    origSeverity="minor"
    prevSeverity="minor"
    rule="fw-pool-empty"
    severity="minor"
    tags=""
    type="equipment"/>
  </outFault>
</faultResolveFault>

```

## IsClone

The IsClone method clones a service profile. The new service profile has the same values as the specified service profile.

## Request Syntax

```

<xs:element name="IsClone" type="IsClone" substitutionGroup="externalMethod"/>
  <xs:complexType name="IsClone" 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:complexType>

```

```

    <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>

```

## Examples

Two examples are provided: cloning a service profile and closing a service template.

### Request (service profile)

```

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

```

### Response (service profile)

```

<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"
    </lsServer>
  </outConfig>
</lsClone>

```

```

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>

```

### Request (service template)

```

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

```

### Response (service template)

```

<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>

```

## IsInstantiateNNamedTemplate

For a specified service template, the IsInstantiateNNamedTemplate method instantiates as many service profiles as are specified in the namedSet parameter.

- dn—Specifies the service template used for instantiating.
- nameSet—Contains the names of the service profiles to be instantiated.
- targetOrg—Specifies the organization in which these service profiles are instantiated.

## Request Syntax

```

<xs:element name="IsInstantiateNNamedTemplate" type="IsInstantiateNNamedTemplate"
substitutionGroup="externalMethod" />
  <xs:complexType name="IsInstantiateNNamedTemplate" 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>

```

## Examples

### 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=""
    >
  </outConfigs>
</lsInstantiateNNamedTemplate>

```

```

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>
```

## Examples

### 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:complexType>

```

```

</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>

```

## Examples

### 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>

```

## IsResolveTemplates

The IsResolveTemplates method retrieves the service profile templates from the specified organization, which is matched hierarchically. The search can be further refined by providing standard querying filters in addition to querying by template-type and a flag to exclude-if-bounded.

Template type can be “initial-template” or “updating-template”.

## Request Syntax

```

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

```

```

</xs:all>
<xs:attribute name="inExcludeIfBound">
  <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="inType">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="initial-template"/>
      <xs:enumeration value="updating-template"/>
      <xs:enumeration value="all"/>
    </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="lsResolveTemplates" type="lsResolveTemplates"
substitutionGroup="externalMethod"/>
<xs:complexType name="lsResolveTemplates" mixed="true">
  <xs:all>
    <xs:element name="outConfigs" type="configMap" minOccurs="0">
      <xs:unique name="unique_map_key_10">
        <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>

```

## Examples

### Request

```
<lsResolveTemplates
  dn="org-root/org-level1"
  cookie="<real_cookie>"
  inExcludeIfBound="false"
  inType="initial-template"
  inHierarchical="no">
</lsResolveTemplates>
```

### Response

```
<lsResolveTemplates
  dn="org-root/org-level1"
  cookie="<real_cookie>"
  response="yes">
  <outConfigs>
    <pair key="ls-service-template-001">
      <lsServer agentPolicyName=" "
        assignState="unassigned"
        assocState="unassociated"
        biosProfileName=" "
        bootPolicyName=" "
        configQualifier=" "
        configState="not-applied"
        descr=" "
        dn="org-root/ls-service-template-001"
        type="initial-template"
        usrLbl=" "
        uuid="derived"
        uuidSuffix="0000-000000000000"
        vconProfileName=" "/>
    </pair>
  </outConfigs>
</lsResolveTemplates>
```

## IsTemplatise

The IsTemplatise 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:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
```

```

        <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>

```

## Examples

### 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>

```

## orgResolveElements

Using a specified organization, policy class ID, and name, the orgResolveElements method resolves the instance of the policy class.

### Request Syntax

```

<xs:element name="orgResolveElements" type="orgResolveElements"
substitutionGroup="externalMethod"/>
  <xs:complexType name="orgResolveElements" 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: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: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="orgResolveElements" type="orgResolveElements"
substitutionGroup="externalMethod"/>
    <xs:complexType name="orgResolveElements" mixed="true">
        <xs:all>
            <xs:element name="outConfigs" type="configMap" minOccurs="0">
                <xs:unique name="unique_map_key_11">
                    <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>

```

## Examples

### Request

```

<orgResolveElements
  dn="org-root/org-Soda"
  cookie="<real_cookie>"
  commCookie="7/15/0/19"

```

```

srcExtSys="10.193.33.221"
destExtSys="10.193.33.221"
srcSvc="sam_extXMLApi"
destSvc="policy-mgr_dme"
inClass="policyPolicySet"
inSingleLevel="no"
inHierarchical="no">
<inFilter>
</inFilter>
</orgResolveElements>

```

## Response

```

<orgResolveElements
  dn="org-root/org-Soda"
  cookie="<real_cookie>"
  commCookie="7/15/0/19"
  srcExtSys="10.193.33.221"
  destExtSys="10.193.33.221"
  srcSvc="sam_extXMLApi"
  destSvc="policy-mgr_dme"
  response="yes"
  errorCode="0"
  errorDescr="">
<outConfigs>
  <pair key="pset-default">
    <policyPolicySet
      adminState="enabled"
      descr="The default Policy Set"
      dn="org-root/pset-default"
      intId="10082"
      name="default" />
  </pair>
  <pair key="pset-myPolicySet3">
    .
  </pair>
  <pair key="pset-policySetSanity">
    <policyPolicySet
      adminState="enabled"
      descr=""
      dn="org-root/org-Soda/pset-policySetSanity"
      intId="24627"
      name="policySetSanity" />
  </pair>
  <pair key="pset-pci_compliance_f">
    <policyPolicySet
      adminState="enabled"
      descr=""
      dn="org-root/pset-pci_compliance_f"
      intId="24539"
      name="pci_compliance_f" />
  </pair>
  <pair key="pset-pci_compliance_h">
    <policyPolicySet
      adminState="enabled"
      descr=""
      dn="org-root/pset-pci_compliance_h"
      intId="24541"
      name="pci_compliance_h" />
  </pair>
</outConfigs>
</orgResolveElements>

```

## 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:complexType>
```

```

</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>

```

## Examples

### 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>

```

## statsClearInterval

The statsClearInterval method resets the collection interval timer for the statsClass. All of the statistics' implicit properties (for example, min, max, and avg calculations) are reset, and the corresponding history properties are updated. The interval updates restart from 1, and the stats collection is reset.

### Request Syntax

```

<xs:element name="statsClearInterval" type="statsClearInterval"
  substitutionGroup="externalMethod"/>
  <xs:complexType name="statsClearInterval" 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="statsClearInterval" type="statsClearInterval"
substitutionGroup="externalMethod"/>
  <xs:complexType name="statsClearInterval" 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>

```

## Examples

### Request

```

<statsClearInterval
  cookie="<real_cookie">
  <inDns>
    <dn value="sys/chassis-1/blade-1/board/temp-stats"/>
  </inDns>
</statsClearInterval>

```

### Response

```

<statsClearInterval
  cookie="<real_cookie">
  response="yes"
  errorCode="0"
  errorDescr="">
</statsClearInterval>

```

## statsResolveThresholdPolicy

The statsResolveThresholdPolicy method resolves threshold policy based on the container class ID. The container class is objects with policies (for example, server domain, lan cloud, san cloud, nas cloud, etc.). The Cisco UCS uses the hierarchy of an organization to resolve the names of policies.

## Request Syntax

```

<xs:element name="statsResolveThresholdPolicy" type="statsResolveThresholdPolicy"
substitutionGroup="externalMethod"/>
  <xs:complexType name="statsResolveThresholdPolicy" 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

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

## Examples

### Request

```
<statsResolveThresholdPolicy
  dn="sys/chassis-1/blade-5/adaptor-1/ext-eth-1/eth-port-stats-rx"
  cookie="<real_cookie>">
</statsResolveThresholdPolicy>
```

### Response

```
<statsResolveThresholdPolicy
  dn="sys/chassis-1/blade-3/adaptor-1/ext-eth-1/eth-port-stats-rx"
  cookie="<real_cookie>"
  response="yes">
  <outConfig>
    <statsThresholdPolicy
      childAction="deleteNonPresent"
      descr="" dn="fabric/lan/thr-policy-default"
      intId="20243"
      name="default" >
      <statsThresholdClass
        childAction="deleteNonPresent"
        descr=""
        intId="32274"
        name="" rn="adaptorEthPortStats"
        statsClassId="adaptorEthPortStats" >
      <statsThr64Definition
        childAction="deleteNonPresent"
        descr=""
        intId="32275"
        name=""
        normalValue="1"
        propId="adaptorEthPortStatstotalPacketsDelta"
        propType="uint64"
        rn="adaptorEthPortStatstotalPacketsDelta" />
      </statsThresholdClass>
    </statsThresholdPolicy>
  </outConfig>
</statsResolveThresholdPolicy>
```





# CHAPTER 4

## Cisco UCS XML Object-Access Privileges

This chapter provides details on the object-access privileges for the Cisco UCS XML API.

This chapter contains the following sections:

- [Privileges Summary Table, page 4-1](#)
- [Privileges Description and Object List, page 4-2](#)

### 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.

[Table 4-1](#) lists each privilege and the initial default user role that has been given that privilege.

**Table 4-1** Summary of Privileges

Internal Name	Label	Description	Default Role Assignment
<a href="#">aaa</a>	AAA	System security and AAA	AAA Administrator
<a href="#">admin</a>	ADMIN	Access to everything (combines all roles)	Administrator
<a href="#">ext-lan-config</a>	EXT_LAN_CONFIG	Configuration of network end points, UCDs, etc.	Network Administrator
<a href="#">ext-lan-policy</a>	EXT_LAN_POLICY	External network policies	Network Administrator
<a href="#">ext-lan-qos</a>	EXT_LAN_QOS	External LAN QoS	Network Administrator
<a href="#">ext-lan-security</a>	EXT_LAN_SECURITY	External LAN security	Network Administrator
<a href="#">ext-san-config</a>	EXT_SAN_CONFIG	Configuration of network end points, UCDs, etc.	Storage Administrator
<a href="#">ext-san-policy</a>	EXT_SAN_POLICY	External SAN policy	Storage Administrator
<a href="#">ext-san-qos</a>	EXT_SAN_QOS	External SAN QoS	Storage Administrator
<a href="#">ext-san-security</a>	EXT_SAN_SECURITY	External SAN security (VACLs, etc.)	Storage Administrator
<a href="#">fault</a>	FAULT	Alarms, alarm policies, etc.	Operations
<a href="#">ls-config</a>	LS_CONFIG	Service profile configuration	Server Profile Administrator

Table 4-1 Summary of Privileges (continued)

Internal Name	Label	Description	Default Role Assignment
ls-config-policy	LS_CONFIG_POLICY	Service profile configuration policy	Server Profile Administrator
ls-ext-access	LS_EXT_ACCESS	Service profile end point access	Server Profile Administrator
ls-network	LS_NETWORK	Service profile network	Network Administrator
ls-network-policy	LS_NETWORK_POLICY	Setting up MAC pools, etc.	Network Administrator
ls-power	LS_POWER	LS power management	Facility Manager
ls-qos	LS_QOS	Service profile QoS	Network Administrator
ls-qos-policy	LS_QOS_POLICY	Setting up ls-level QoS	Network Administrator
ls-security	LS_SECURITY	Service profile security	Server Security Administrator
ls-security-policy	LS_SECURITY_POLICY	Setting up security policies	Server Security Administrator
ls-server	LS_SERVER	Service profile server management	Server Security Administrator
ls-server-oper	LS_SEVER_OPER	Server profile consumer role	Server Profile Administrator
ls-server-policy	LS_SERVER_POLICY	Service profile pool policy	Server Security Administrator
ls-storage	LS_STORAGE	Service profile storage	Storage Administrator
ls-storage-policy	LS_STORAGE_POLICY	Service profile storage policy	Storage Administrator
operations	OPERATIONS	Logs, call home functionality, etc.	Operations
pn-equipment	PN_EQUIPMENT	Server hardware management	Server Equipment Administrator
pn-maintenance	PN_MAINTENANCE	Server maintenance (update BIOS, etc.)	Server Equipment Administrator
pn-policy	PN_POLICY	Physical server policies	Server Equipment Administrator
pn-security	PN_SECURITY	Physical node security	Server Equipment Administrator
pod-config	POD_CONFIG	Pod configuration	Network Administrator
pod-policy	POD_POLICY	Pod policies	Network Administrator
pod-qos	POD_QOS	Internal pod-QoS (if needed)	Network Administrator
pod-security	POD_SECURITY	Pod security	Network Administrator
power-mgmt	POWER_MGMT	Data center power management	Facility Manager
read-only	READ_ONLY	Read-only access	Available to all roles

## Privileges Description and Object List

This section describes each of the available 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:XmlCICConnPolicy, comm:XmlCICConnPolicy, 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

## Is-ext-access

**Purpose:** Service profile end point access

**Responsible role:** Server Profile Administrator

This privilege is not used.

## Is-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

## Is-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

## Is-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

## Is-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

## Is-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

## Is-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:LocalDiskConfigDef, storage: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:Smtpt, 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:VfIntelVTFForDirectedIO, 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.

## 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.

## Power Management

This section describes power management privileges. The facility manager is responsible for providing and ensuring availability of power for the data center and all contents.

### 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

### 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-power

**Purpose:** Service profile power management

**Responsible role:** Facility Manager



## INDEX

---

### A

- all bits filter
  - example [2-12](#)
- AND filter
  - example [2-11](#)
- any bits filter
  - example [2-11](#)
- audience [i-xi](#)
- authentication methods
  - description [1-6](#)
  - examples [3-1](#)

---

### C

- composite filters
  - description [1-8](#)
  - example [2-11](#)
- computeBlade attribute, usage [2-13, 2-14](#)
- configFindDnsByClassId
  - example [2-4](#)
- configResolveChildren
  - example [2-4](#)
- configResolveClasses
  - example [2-5](#)
- configResolveDn
  - example [2-5](#)
- configResolveDns
  - example [2-5](#)
- configResolveParent
  - example [2-5](#)
- configuration changes, making [3-1](#)
- configuration methods [1-8](#)

cookie [2-1](#)

---

### D

- data management engine
  - place in UCS API flow [1-3](#)
- distinguished name
  - description [1-5](#)
  - resolving [2-5](#)
- distinguished names
  - resolving [2-5](#)

---

### E

- empty results [1-11](#)
- event subscription
  - eventSubscribe method [1-9](#)
- examples
  - querying a MAC pool [2-6](#)
  - resolving a distinguished name [2-5](#)
  - resolving children [2-4](#)
  - resolving distinguished names [2-5](#)
  - resolving parents [2-5](#)

---

### F

- failed request [1-11](#)
- faults, querying [2-8](#)
- filter examples
  - all bits filter [2-12](#)
  - AND, OR, NOT composite filter [2-11](#)
  - any bits filter [2-11](#)
  - greater than filter [2-10](#)

greater than or equal to filter [2-10](#)  
 inequality filter [2-10](#)  
 less than filter [2-11](#)  
 less than or equal to filter [2-11](#)  
 not equal to filter [2-11](#)  
 NOT filter [2-14](#)

## filters

composite [1-8](#)  
 modifier [1-8](#)  
 property [1-7](#)  
 simple [1-7](#)

## G

greater than filter  
   example [2-10](#)  
 greater than or equal to filter  
   example [2-10](#)

## H

HTTP in UCS API communication [1-1](#)  
 HTTPS in UCS API communication [1-1](#)

## I

inequality filter  
   example [2-10](#)

## L

less than filter  
   example [2-11](#)  
 less than or equal to filter  
   example [2-11](#)

## M

MAC pool  
   querying [2-6](#)  
 macpoolAddr class, usage [2-7](#)  
 management information tree (MIT)  
   place in UCS API flow [1-3](#)  
   structure [1-3](#)  
 modifier filters description [1-8](#)

## N

NOT filter  
   example [2-14](#)

## O

OR filter  
   example [2-11](#)

## P

packet analyzer, for UCS GUI XML interchange [1-9](#)  
 property filters description [1-7](#)

## Q

querying  
   faults [2-8](#)  
   filters description [1-7](#)  
   finding DNs by class ID [2-4](#)  
   MAC pool [2-6](#)  
   obtaining classes [2-5](#)  
   statistics [2-7](#)  
 query methods [1-6](#)  
   configFindDnsByClassId [2-4](#)  
   configResolveChildren [2-4](#)  
   configResolveClasses [2-5](#)

[configResolveDn](#) [2-5](#)  
[configResolveDns](#) [2-5](#)  
[configResolveParent](#) [2-5](#)  
[query method usage](#) [2-3](#)

---

## R

[relative name](#)  
     [example](#) [1-5](#)  
[resolving children example](#) [2-4](#)  
[resolving parents](#)  
     [example](#) [2-5](#)

---

## S

[simple filters description](#) [1-7](#)  
[statistics, querying](#) [2-7](#)

---

## U

[UCS API](#)  
     [empty results example](#) [1-11](#)  
     [failed request example](#) [1-11](#)  
     [flow](#) [1-3](#)  
     [model documentation](#) [1-2](#)  
     [response to success or failure example](#) [1-9](#)  
[UCS Manager](#)  
     [information model](#) [1-2](#)  
[Unified Computing System \(UCS\)](#)  
     [overview](#) [1-2](#)  
     [XML API](#) [1-1](#)

---

## W

[wildcard filter](#)  
     [example](#) [2-10](#)

---

## X

[XML](#)  
     [capturing GUI-UCS traffic](#) [1-9](#)  
     [UCS API flow](#) [1-3](#)

