Cisco UCS Central XML API

This chapter includes the following sections:

Cisco UCS Central Overview

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

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

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

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

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

Cisco UCS Central Features

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

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

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

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

Domain Groups

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

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

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

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

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

Policies

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

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

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

Policy Browser

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

The following features are supported:

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

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

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

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

Pools

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

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

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

Information Sharing between UCS Domains and Cisco UCS Central

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

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

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

Introduction to the Cisco UCS Central XML API

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

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

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

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

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

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

The API model includes the following programmatic entities:

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

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

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

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

Management Information Model

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

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

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

Illustration of MIM Structure Showing Five Chassis

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

Central Manager

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

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

Cisco UCS Central contains the following service provider DMEs:

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

Cisco UCS Central XML API Sample Flow

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

Object Naming

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

Distinguished Name

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

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

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

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

Relative Name

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

For example, this distinguished name:

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

is composed of the following relative names:

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

API Method Categories

Each method corresponds to an XML document.

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

Authentication Methods

Authentication methods authenticate and maintain the session. For example:

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

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

Query Methods

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

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

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

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

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

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

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

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

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

Simple Filters

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

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

Property Filters

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

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

Composite Filters

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

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

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

Modifier Filter

A modifier filter changes the results of a contained filter.

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

Configuration Methods

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

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

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

Event Subscription Methods

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

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

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

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

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

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

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

Capturing XML Interchange Between the GUI and Cisco UCS Central

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

Success or Failure Response

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

Successful Requests

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

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

Failed Requests

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

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

Empty Results

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

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

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