Cisco CIMC XML API

This chapter includes the following sections:

About the Cisco CIMC XML API

The Cisco CIMC XML API is a programmatic interface to the Cisco Integrated Management Controller (CIMC) software for a C-Series Rack-Mount Server. 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 CIMC is stored in a hierarchical tree structure known as the MIT (Management Information Tree), which is completely accessible through the XML API.

The Cisco CIMC XML API implements a subset of the methods and management information model available in the Cisco UCS Manager XML API. The behavior of both APIs is similar in syntax and semantics, and you can use the same client development tools and techniques for both. The scope of the Cisco CIMC XML API is limited to a single C-Series Rack-Mount Server, in contrast to the Cisco UCS Manager XML API, which controls an entire Cisco UCS domain consisting of switches, FEX modules, servers, and other devices.

Using the Cisco CIMC XML API, the user has programmatic access to CIMC to configure, administer, and monitor the server. The API provides most of the functions that are accessible through the CIMC CLI and GUI interfaces.

Operation of the API is transactional and terminates on a single data model maintained in CIMC.

The API model includes the following programmatic entities:

- Classes—Define the properties and states of objects in the MIT.
- 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 CIMC 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 MIT. This complete operation is done in a single transaction.

Event subscription is supported. Up to four Cisco CIMC XML API clients can subscribe to receive event notifications from CIMC. The event subscription operation establishes a connection session allowing a client to receive XML-formatted event notification messages that are sent asynchronously by CIMC.

Note In Release 1.5(1.x), the Cisco CIMC XML API sends event notifications only for fault-related events.

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 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 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 CPUs, DIMMs, adapter cards, fans, and power supply units..

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).

CIMC Management Information Model

The CIMC management information model is a subset of the Cisco UCS management information model. A C-Series Rack-Mount Server is modeled starting with `sys/rack-unit-1` in the MIT as in the following example:

Figure 1: Illustration of the CIMC MIM Structure

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

|———sys———————————————— (sys)
     |———rack-unit-1————————————(sys/rack-unit-1)
          |———adaptor-1————————————(sys/rack-unit-1/adaptor-1)
          |———psu-1————————————(sys/rack-unit-1/psu-1)
          |———psu-2————————————(sys/rack-unit-1/psu-2)
```

Cisco CIMC XML API Sample Flow

A typical request comes into CIMC 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.

The following figure shows how CIMC processes a boot server request. The following table describes the steps involved in a boot server request.

Figure 2: Sample Flow of Boot Server Request

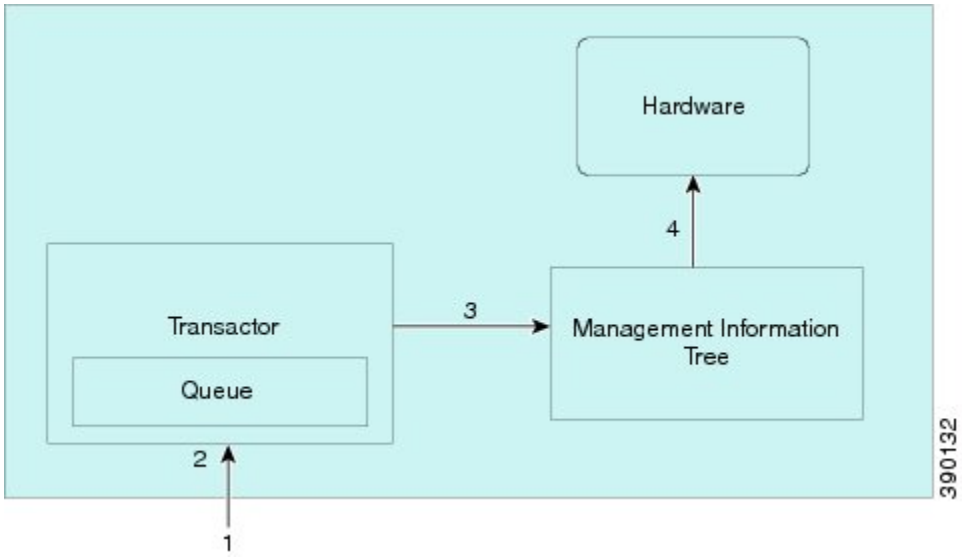

Table 1: Explanation of Boot Server Request

Step	Command/Process	Operational Power State of MO (Server)
1	CMD request: boot server	Down
2	Request queued	Down
3	State change in management information tree and make persistent the managed object (MO) state change	Down
4	Apply boot stimuli	Up

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/rack-unit-1/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/rack-unit-1/adaptor-1/host-eth-2"/>
```

is composed of the following relative names:

```
topSystem MO: rn="sys"
computeRackUnit MO: rn ="rack-unit-1"
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 4 sessions to the Cisco UCS can be opened at any one time.

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

Tip In CIMC, HTTP to HTTPS redirection is enabled by default. To capture HTTP packets between the client application and CIMC, disable redirection in the CIMC GUI or CLI.

Query Methods

Query methods obtain information on the current configuration state of an object. The following are query methods supported in Release 1.5(1.x):

- configResolveDn—Retrieves objects by DN.
- configResolveClass—Retrieves objects of a given class.
- configResolveChildren—Retrieves the child objects of an object.
- configResolveParent—Retrieves the parent object of an object.

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 CIMC 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 CIMC configuration. The resources required for CIMC to process the request can be high, causing CIMC 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).

Note Until a host is powered on at least once, CIMC may not have complete inventory and status information. For example, if CIMC 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.

Configuration Methods

The Cisco CIMC XML API supports only a single method to make configuration changes to managed objects:

- configConfMo—Affects a single managed object (for example, a DN).

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, CIMC starts sending all new events as they occur. You can unsubscribe from these events using the eventUnsubscribe method.

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.

An event channel connection opened by a user will be closed automatically by CIMC after 600 seconds of inactivity associated with the event channel session cookie. To prevent automatic closing of the event channel connection by CIMC, 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 CIMC using the same event channel session cookie.

Note In Release 1.5(1.x), the Cisco CIMC XML API sends event notifications only for fault-related events.

Success or Failure Response

When CIMC 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 Response

When a request has executed successfully, CIMC 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/rack-unit-1/adaptor-2/ext-eth-0`:

```
<configResolveDn
     dn="sys/rack-unit-1/adaptor-2/ext-eth-0"
     cookie="<real_cookie>"
     inHierarchical="false"/>
```

The response includes the following information:

```
<configResolveDn
    cookie="<real_cookie>"
    response="yes"
    dn="sys/rack-unit-1/adaptor-2/ext-eth-0">
    <outConfig>
        <adaptorExtEthIf
            id="0"
            ifType="physical"
            linkState="up"
            mac="00:22:BD:D6:42:DA"
            name=""
            operState="up"
            portId="0"
            purpose="general"
            transport="CE"
            type=""
            dn="sys/rack-unit-1/adaptor-2/ext-eth-0" >
        </adaptorExtEthIf>
    </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="sys/rack-unit-1/adaptor-1/ext-eth-0"
     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 CIMC. If the object does not exist, CIMC 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
    cookie="<real_cookie>"
    response="yes"
    dn="sys/rack-unit-1/adaptor-9999">
    <outConfig>
```

```
    </outConfig>
</configResolveDn>
```