Using the Cisco IMC XML API Methods

This chapter includes the following sections:

- Authentication Methods, on page 1
- Query Methods, on page 3

Authentication Methods

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

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

Login

To log in, the XML API client establishes a TCP connection to the Cisco IMC HTTP (or HTTPS) server and posts an XML document containing the `aaaLogin` method.

In the following example, the Telnet utility is used to establish a TCP connection to port 80 of the Cisco IMC with IP address 192.0.20.72. The path used is /nuova.

```
$ telnet 192.0.20.72 80
POST /nuova HTTP/1.1
USER-Agent: lwp-request/2.06
HOST: 192.0.20.72
Content-Length: 62
Content-Type: application/x-www-form-urlencoded
```

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

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

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 IMC reads the request and performs the actions as provided in the method. Cisco IMC 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="admin">
6 </aaaLogin>
```

Each line in the response should be interpreted as follows:

1. Specifies the method used to login.
2. Confirms that this is a response.
3. Provides the session cookie.
4. Specifies the recommended cookie refresh period. The default login session length is 600 seconds.
5. Specifies the privilege level assigned to the user account (this can be admin, user, or readonly).
6. Closing tag.

Alternatively, you can use the cURL utility to log in to the XML API, as shown in the following example:

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

If HTTPS is enabled, you must use HTTPS in the cURL command, as shown in the following example:

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

Refreshing the Session

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

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

Logging Out of the Session

Use the following method to log out of a session:

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

Unsuccessful Responses

Failed login:

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

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

```
<configResolveDn
    cookie="<real_cookie>"
    response="yes"
    dn="sys/rack-unit-1/adaptor-9999">
    <outConfig>
    </outConfig>
</configResolveDn>
```

Bad request:

```
<configConfMo
    cookie="<real_cookie>"
    response="yes"
    dn="sys/rack-unit-1/adaptor-1/ext-eth-0">
    errorCode="103"
    invocationResult="unidentified-fail"
    errorDescr="can't create; object already exists.">
</configConfMo>
```

Query Methods

Using `configResolveChildren`

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

- This method obtains all child objects of a named object that are instances of the named class. If a class name is omitted, all child objects of the named object are returned.

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

See the example request/response in configResolveChildren.

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 adapters, use **`adaptorUnit`** as the attribute value for classId in the query. This example queries for all instances of the `adaptorUnit` class:

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

See the example request/response in configResolveClass.

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.

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.