Perform Manual XML API Calls to CIMC

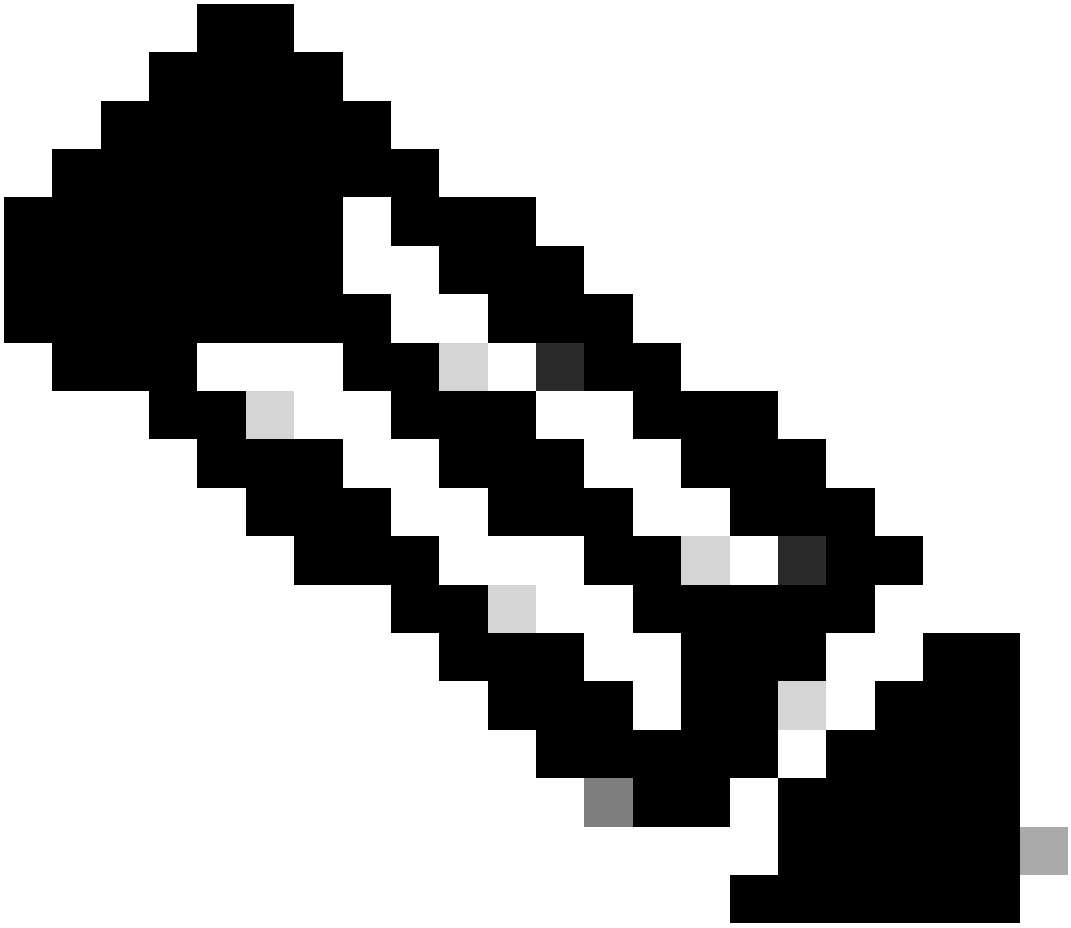
Contents

Introduction

This document describes how to perform manual XML API Calls to Cisco Integrated Management Controller (CIMC).

Components Used

- A Linux machine (any distribution).
- Network connectivity between the Linux machine and Cisco CIMC (a successful ping test is sufficient).



Note: The file names (main.sh, login.xml, get_summary.xml) used in this guide are arbitrary. You are free to use your own naming conventions, as long as they are properly referenced throughout the process.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

In this demonstration, a Bash script named "**main.sh**" is used to execute API calls to the server.

Please note that alternative methods, such as CURL(directly from CLI) or Python, can also be to achieve this same outcome.

Perform XML API Calls to CIMC

The CIMC XML API requires an initial authentication step using an "**aaaLogin**" API call. During this

process, a session cookie is retrieved, which is then used to authenticate all subsequent API calls.

Step 1

Start by creating a file named **main.sh** in the remote Linux machine. This file contains a bash script used to send the API calls.

The **main.sh** script is executed multiple times throughout this exercise: first for authentication and then for retrieving information from the CIMC in subsequent calls.

Here is the content of the **main.sh** file:

```
#!/bin/bash
# CONFIGURATION - Edit these variables
CIMC_IP="X.X.X.X" # <<<<<<<< Customer CIMC IP address
USERNAME="admin"
PASSWORD="xxxxxxxx" # <<<<<<< CIMC Password in plaintext
XML_PAYLOAD_FILE="login.xml" # <<<<<<< Referencing the login.xml file for authentication and cookie ret
CIMC_URL="https://${CIMC_IP}/nuova" # <<<<<<< Call is made to this URL

# Check if payload file exists
if [ ! -f "$XML_PAYLOAD_FILE" ]; then
echo "Error: XML payload file '$XML_PAYLOAD_FILE' not found."
exit 1
fi
# Send XML request using curl
curl -k -s -u "${USERNAME}:${PASSWORD}" -H "Content-Type: text/xml" -d @"${XML_PAYLOAD_FILE}" "${CIMC_U
```

The bash script defines the CIMC login parameters and also executes an XML API call, specified in another file called**login.xml** identified by a variable called**XML_PAYLOAD_FILE**.

The script also checks whether the file **login.xml**, defined by the variable **XML_PAYLOAD_FILE** exists and is a regular file.

If the file defined by XML_PAYLOAD_FILE does not exist, the script prints an error and exits.

Savethe file and make it executable by running this command in the CLI:

```
$ sudo chmod +x main.sh
```

Step 2

Next, create the file called**login.xml** in the same linux directory as the **main.sh** file.

This login file contains the actual**aaaLogin**XML API call that is sent to the CIMC to retrieve a session cookie. The retrieved cookie is used for subsequent API calls:

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

Remember to replace the CIMC username and password with the appropriate credentials.

Execute the **main.sh** script in CLI to retrieve a cookie:

```
$ sudo ./main.sh
```

If the API call is successful, the XML response returned contains a key called**outCookie**with its value being the retrieved cookie as shown:

```
<?xml version="1.0"?>
<aaaLogin cookie="" response="yes" outCookie="xxxxxxxxx/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" outRefreshPeriod="600" outPriv="admin" outSessionId="18"
outVersion="4.3(5.250001)"> </aaaLogin>
```

From the output, locate the **outCookie** value.

Save this cookie value.

Step 3

Create a new file in the same directory where the **main.sh** file is located. Name the file **get_summary.xml.**

For a comprehensive list of API requests that can be used with the Cisco IMC, refer to the **Cisco UCS Rack-Mount Servers CIMC XML API Programmer's Guide**.

The new **get_summary.xml** file is used to retrieve the **Server Summary Information and Host Power State**. using the XML block of code in the reference documentation, but replacing the **cookie** key value with the earlier retrieve cookie.

```
<configResolveClass cookie="<cookie_value>" inHierarchical="false" classId="computeRackUnit"/>
```

Replace **<cookie_value>** with the**outCookie**value obtained from the earlier **login.xml**response. The updated request looks like this:

```
<configResolveClass cookie="xxxxxxxxx/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
inHierarchical="false" classId="computeRackUnit"/>
```

Be sure to replace the **<cookie_value>** with your actual cookie value retrieved during the authentication process.

Step 4

Once the cookie has been added to the new **get_summary.xml** file, update the **main.sh** script to reference the **"get_summary.xml"** as value for the **XML_PAYLOAD_FILE** variable for this request.

```
#!/bin/bash
# CONFIGURATION - Edit these variables
CIMC_IP="X.X.X.X"
USERNAME="admin"
PASSWORD="xxxxxxxx"
XML_PAYLOAD_FILE="get_summary.xml" # <<<<<<< Referencing the get_summary.xml file for subsequent API ca
CIMC_URL="https://${CIMC_IP}/nuova"
# Check if payload file exists
if [ ! -f "$XML_PAYLOAD_FILE" ]; then
echo "Error: XML payload file '$XML_PAYLOAD_FILE' not found."
exit 1
fi
# Send XML request using curl
curl -k -s -u "${USERNAME}:${PASSWORD}" -H "Content-Type: text/xml" -d @"${XML_PAYLOAD_FILE}" "${CIMC_U
```

Run the **main.sh** script again to execute the updated API request.

```
$ sudo ./main.sh
```

You then receive an XML-formatted API response returning the requested object from CIMC.

```
<?xml version="1.0"?>
<configResolveClass cookie="xxxxxxxxx/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" response="yes" classId="computeRackUnit">
<outConfigs>
<computeRackUnit dn="sys/rack-unit-1" adminPower="policy" availableMemory="524288" model="UCSC-C220-M6S" memorySpeed="3200" name="UCS C220
M6S" numOfAdaptors="2" numOfCores="56" numOfCoresEnabled="56" numOfCpus="2" numOfEthHostIfs="0" numOfFcHostIfs="0" numOfThreads="112" operPower
="off" originalUuid="XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" presence="equipped" serverId="1"
serial="XXXXXXXXXX" totalMemory="524288" usrLbl="" uuid="XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" vendor="Cisco Systems
Inc" cimcResetReason="graceful-reboot
" assetTag="Unknown" adaptorSecureUpdate="Enabled" resetComponents="components" storageResetStatus="NA" vicResetStatus="NA" bmcResetStatus="NA" sma
rtUsbAccess="disabled" smartUsbStatus="Disabled" biosPostState="pending"/>
</outConfigs>
</configResolveClass>
```

Troubleshoot

It is important to note that the **aaaLogin** XML API call returns a session cookie with an **outRefreshPeriod** of approximately **600 seconds**. This means, the cookie expires after ten (10) minutes if not refreshed, and a new cookie is required to continue performing API requests.

If you attempt to use an expired cookie (after 600 seconds), a 552 "**Authorization required**" XML error response block is returned:

<?xml version="1.0"?>
<configResolveDn cookie="xxxxxxxxx/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" response="yes" errorCode="552" invocationResult="service-unavailable" errorDescr="Authorization required"> </configResolveDn>