Cisco Identity Services Engine API Reference Guide, Release 1.2

October 2013

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco website at www.cisco.com/go/offices.

Text Part Number: OL-26134-01

Cisco Identity Services Engine API Reference Guide, Release 1.2

Preface

Purpose

This guide provides a developer, system or network administrator, or system integrator basic guidelines for using the outlined APIs in a Cisco ISE deployment.

The External RESTful Services API and related API calls can be used to perform CRUD (Create, Read, Update, Delete) operations on Cisco ISE resources. The External RESTful Services API is based on HTTP protocol and REST methodology.

Note This guide is not intended to replace *Cisco Identity Services Engine User Guide, Release 1.2*. For more information about a Cisco ISE network, its nodes and personas, concepts of operation or usage, and how to use the Cisco ISE user interface, see *Cisco Identity Services Engine User Guide, Release 1.2.*

Audience

This guide is intended for experienced network administrators who administer Cisco ISE appliances, system integrators who want to make use of the APIs, and third-party partners who are responsible for managing and troubleshooting Cisco ISE deployments. As a prerequisite to using it, you should have a basic understanding of troubleshooting and diagnostic practices and how to make and interpret API calls.

Document Organization

- Part 1—Monitoring REST APIs
  - Chapter 1, “Introduction to the Monitoring REST API”

Document Conventions

Note Means *reader take note*. Notes contain helpful suggestions or references to materials not contained in the manual.

Convention	Item
bold	Commands, keywords, and user-entered text as well as tab and button names in procedural text appear in **bold** font.
italic	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
`courier`	Terminal sessions and information the system displays is in courier (monospace, fixed-width) font.
>	A right-angle bracket (>) surrounded by spaces is used to separate options in a menu path.
< >	In examples that do not allow italics, such as ASCII output, arguments for which you must supply a value appear in angle brackets.

Related Documentation

- Release-Specific Documentation, page ix
- Platform-Specific Documentation, page ix

Note Updates to printed and electronic documentation can be found at http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html.

Release-Specific Documentation

Table 1 Product Documentation for Cisco Identity Services Engine

Document Title	Location
Release Notes for the Cisco Identity Services Engine, Release 1.2	http://www.cisco.com/en/US/products/ps11640/prod_release_notes_list.html
Cisco Identity Services Engine Network Component Compatibility, Release 1.2	http://www.cisco.com/en/US/products/ps11640/products_device_support_tables_list.html
Cisco Identity Services Engine User Guide, Release 1.2	http://www.cisco.com/en/US/products/ps11640/products_user_guide_list.html
Cisco Identity Services Engine Hardware Installation Guide, Release 1.2	http://www.cisco.com/en/US/products/ps11640/prod_installation_guides_list.html
Cisco Identity Services Engine, Release 1.2 Migration Tool Guide	http://www.cisco.com/en/US/products/ps11640/prod_installation_guides_list.html
Cisco Identity Services Engine Sponsor Portal User Guide, Release 1.2	http://www.cisco.com/en/US/products/ps11640/products_user_guide_list.html
Cisco Identity Services Engine CLI Reference Guide, Release 1.2	http://www.cisco.com/en/US/products/ps11640/prod_command_reference_list.html
Cisco Identity Services Engine Troubleshooting Guide, Release 1.2	http://www.cisco.com/en/US/products/ps11640/prod_troubleshooting_guides_list.html
Regulatory Compliance and Safety Information for Cisco Identity Services Engine, Cisco 1121 Secure Access Control System, Cisco NAC Appliance, Cisco NAC Guest Server, and Cisco NAC Profiler	http://www.cisco.com/en/US/products/ps11640/prod_installation_guides_list.html
Cisco Identity Services Engine In-Box Documentation and China RoHS Pointer Card	http://www.cisco.com/en/US/products/ps11640/products_documentation_roadmaps_list.html

Platform-Specific Documentation

- Cisco ISE
  http://www.cisco.com/en/US/products/ps11640/index.html
- Cisco Secure Access Control Server (ACS)
  http://www.cisco.com/en/US/products/ps9911/tsd_products_support_series_home.html
- Cisco NAC Appliance
  http://www.cisco.com/en/US/products/ps6128/tsd_products_support_series_home.html
- Cisco NAC Profiler
  http://www.cisco.com/en/US/products/ps8464/tsd_products_support_series_home.html
- Cisco NAC Guest Server
  http://www.cisco.com/en/US/products/ps10160/tsd_products_support_series_home.html

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see *What's New in Cisco Product Documentation* at: http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html.

Subscribe to *What's New in Cisco Product Documentation*, which lists all new and revised Cisco technical documentation, as an RSS feed and deliver content directly to your desktop using a reader application. The RSS feeds are a free service.

PART 1

Monitoring REST API

Introduction to the Monitoring REST API

The Monitoring REST API allows allow you to gather session and node-specific information by using Monitoring nodes in your network. A session is defined as the duration between when you access a desired node and complete the operations needed to gather information.

The following Monitoring REST API categories are supported in Cisco ISE, Release 1.2:

- Session Management
- Troubleshooting
- Change of Authorization (CoA)

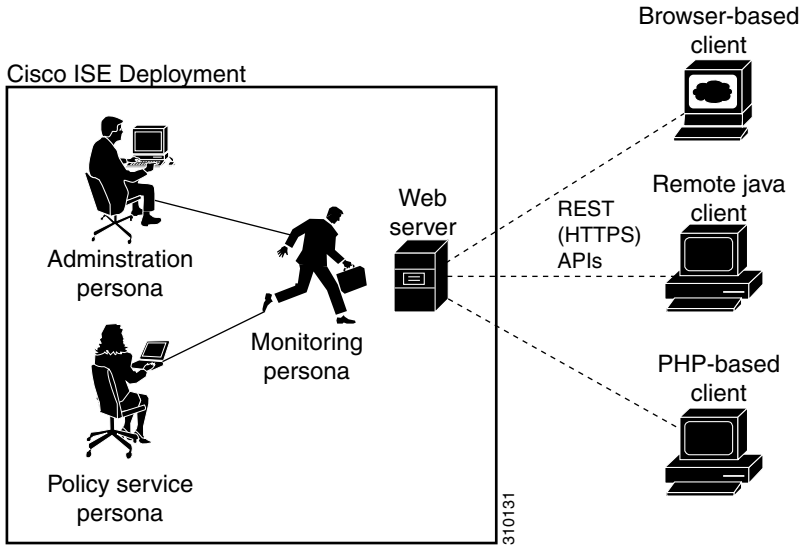
Note You use only supported categories to gather information about endpoints being monitored by the Monitoring persona. Monitoring is one of three supported personas that a node type can perform in a Cisco ISE, Release 1.2, deployment. For the remainder of this guide, "Monitoring node" will be used to describe the Monitoring persona of a Cisco ISE node.

Any attempt to use these categories to gather information about the Policy Service persona of a Cisco ISE appliance will result in an error. For more information about Cisco ISE nodes and personas, see *Cisco Identity Services Engine User Guide, Release 1.2*.

Monitoring REST API calls allow you to locate, monitor, and accumulate important real-time, session-based information stored in individual endpoints in a network. You can access this information through a Monitoring node.

The real-time, session-based information that you gather can help understand Cisco ISE operations and assist in diagnosing conditions or issues. It can also be used to troubleshoot error conditions or an activity or behavior that may be affecting monitoring operations. As shown in Figure 1-1, the Monitoring REST API calls are used to access the Monitoring node and retrieve important session-based information that is stored in the Cisco ISE deployment endpoints.

Figure 1-1 Monitoring REST API Calls in a Distributed Deployment



Verifying a Monitoring Node

Before you Begin

Before you can successfully invoke the API calls on a Monitoring node, you need to verify that the node you want to monitor is valid.

Note To be able to use a public Monitoring REST API, you must first authenticate with Cisco ISE using valid credentials.

Step 1 Enter valid login credentials (Username and Password) in the Cisco ISE Login window, and click **Login**.

The Cisco ISE dashboard and user interface appears.

Step 2 Choose **Authorization > System > Deployment**.

The Deployment Nodes page appears, which lists all configured nodes that are deployed.

Step 3 In the Roles column of the Deployment Nodes page, verify that the role for the target node that you want to monitor is listed as a Monitoring node.

Supported API Calls

The following tables describe the different types of API calls and provide an example of the API call format:

- Table 1-1 on page 1-3—defines API calls for session management.
- Table 1-2 on page 1-6—defines API calls for troubleshooting.
- Table 1-3 on page 1-7—defines CoA API calls.

If you intend to use a generic programmatic interface to authenticate with the Monitoring REST API supported by Cisco ISE, you need to first create a REST-based client that bridges Cisco ISE and the specific tool you use. You then use this REST client to authenticate with the Cisco ISE Monitoring REST APIs, marshal and submit the API requests to the Monitoring nodes, and then unmarshal the API responses and pass them on to the specified tool.

Table 1-1 Cisco ISE Session Management API Calls

API Call Category	Description and Example
Session Counters	
ActiveCount	Lists the number of active sessions. https://<ISEhost>/ise/mnt/Session/ActiveCount
PostureCount	Lists the number of Postured endpoints. https://<ISEhost>/ise/mnt/Session/PostureCount **Note** Posture is a service that aids in checking the state (or posture) for all the endpoints that connect to a Cisco ISE network. Cisco ISE utilizes NAC Agent for checking the posture compliance of a device.
ProfilerCount	Lists the number of active Profiler service sessions. https://<ISEhost>/ise/mnt/Session/ProfilerCount **Note** Profiler is a service that aids in identifying, locating, and determining the capabilities of all attached endpoints on a Cisco ISE network.

Table 1-1 Cisco ISE Session Management API Calls (continued)

API Call Category	Description and Example
Session List **Note** A session list includes the MAC address, network access device (NAD) IP address, username, and session ID information associated with a session.	
ActiveList	Lists all active sessions. https://<ISEhost>/ise/mnt/Session/ActiveList **Note** In this release of Cisco ISE, the maximum number of active authenticated endpoint sessions that can be displayed is limited to 250,000.
AuthList	Lists all currently active authenticated sessions. https://<ISEhost>/ise/mnt/Session/AuthList/<parameteroptions> You can specify the following parameter options that will return different values: • null/null—Lists all active authenticated sessions. • null/endtime—Lists all active authenticated sessions after the specified end time. • starttime/null—Lists all active authenticated sessions before the specified start time. • starttime/endtime—Lists all active authenticated sessions between the specified start time and end time. Enter the date and time for the start time and end time in the following format: YYYY-MM-DD hh:mm:ss.s where: • YYYY—four-digit year • MM—two-digit month (01=January, and so on) • DD—two-digit day of the month (01 through 31) • hh—two-digit hour (00 through 23) (a.m. and p.m. are not allowed) • mm—two-digit minute (00 through 59) • ss—two-digit second (00 through 59) • s—one or more digits representing a decimal fraction of a second **Note** Every Cisco ISE node is configured with a time zone. Recommended time zone is UTC. See AuthList API Call Data, page 2-8, for samples that show all four parameter options.

Table 1-1 Cisco ISE Session Management API Calls (continued)

API Call Category	Description and Example
Session Attributes **Note** This is a timestamp-based search for the latest session that contains the specified search attribute.	
MACAddress	Searches the database for the latest session that contains the specified MAC address. https://<ISEhost>/ise/mnt/Session/MACAddress/<macaddress> **Note** XX:XX:XX:XX:XX:XX is the MAC address format and is not case sensitive (for example, 0a:0B:0c:0D:0e:0F). **Note** The MAC address serves as the only unique key to finding the correct session you want to monitor. Use the ActiveList API call to list all active sessions and their MAC addresses, from which you can base your MAC address search.
UserName	Searches the database for the latest session that contains the specified username. https://<ISEhost>/ise/mnt/Session/UserName/<username> **Note** Usernames must conform to the same Cisco ISE password policy used for network usernames. The only invalid character for the Monitoring REST APIs is the backslash (\) character. For details, see "User Password Policy" in *Cisco Identity Services Engine User Guide, Release 1.1*.
IPAddress	Searches the database for the latest session that contains the specified NAS IP address. https://<ISEhost>/ise/mnt/Session/IPAddress/<nasipaddress> **Note** xxx.xxx.xxx.xxx is the NAS IP address format (for example, 10.10.10.10).

For specific details about Cisco ISE API calls for session management, see Chapter 2, "Using API Calls for Session Management".

Table 1-2 Cisco ISE Troubleshooting API Calls - Troubleshooting

API Call	Description and Example
Version	Lists the node version and type. https://<ISEhost>/ise/mnt/Version Node type can be any of the following values (0-3): 0—STAND_ALONE_MNT_NODE 1—ACTIVE_MNT_NODE 2—STAND_BY_MNT_NODE 3—NOT_AN_MNT_NODE **Note** STAND_ALONE_MNT_NODE means it is a Monitoring node that does not function in any distributed deployment. ACTIVE_MNT_NODE means it is a primary node in a primary-secondary relationship in a distributed deployment. STAND_BY_MNT_NODE means it is a secondary node in a primary-secondary pair in a distributed deployment. NOT_AN_MNT_NODE means it is not a Monitoring node. See *Cisco Identity Services Engine User Guide, Release 1.1* for details about the supported ISE nodes and personas.
FailureReasons	Lists the reasons for failure. https://<ISEhost>/ise/mnt/FailureReasons Each failure reason displays an error code (failureReason id), a brief description (code), a failure reason (cause), and a possible response (resolution), as shown in the following example: <failureReason id="100009"> <code> 100009 WEBAUTH_FAIL <cause> This may or may not be indicating a violation. <resolution> Please review and resolve this issue according to your organization's policy. **Note** The FailureReasons API call to be called only once to gather the information from the Monitoring node. You should store the contents of any returned failure reasons into your own file system or database. The returned contents of these API calls are intended to be used for reference purposes. If you experience any issues during authentication, you should compare the failure reason code provided in the authentication response with the list of failure reasons that you have stored in your own file system or database. For a complete list of Cisco ISE failure reasons, see Appendix A, "Using the Failure Reasons Report".

Table 1-2 Cisco ISE Troubleshooting API Calls - Troubleshooting (continued)

API Call	Description and Example
AuthStatus	Lists the authentication status for all sessions. https://<ISEhost>/ise/mnt/AuthStatus/MACAddress/<macaddress>/<numberofseconds>/<numberofrecordspermacaddress>/All **Note** The seconds parameter <numberofseconds> is user-configurable, the range is from 0 to 432000 seconds (5 days).
Get Session Accounting Status	
AcctStatus	Lists the accounting status of all sessions within a specific period of time. https://<ISEhost>/ise/mnt/Session/AcctStatusTT/MACAddress/<macaddress>/<numberof seconds> **Note** The seconds parameter <numberofseconds> is user-configurable, with the range is from 0 to 432000 seconds (5 days).

For specific details about Cisco ISE API calls for troubleshooting, see Chapter 2, "Using API Calls for Session Management".

Table 1-3 Cisco ISE Change of Authorization API Calls

API Call	Description and Example
Reauth	Sends a session reauthentication command and type. https://<ISEhost>/ise/mnt/CoA/Reauth/<serverhostname>/<macaddress>/<reauthtype>/<nasipaddress>/<destinationipaddress> Where <*ISEhost*> denotes the ip address of the ISE host, <*serverhostname*> denotes the name of the ISE server, <*nasipaddress*> denotes the identifying ip address of NAS, and <*destinationipaddress*> denotes the ip address of the destination. Reauth type can be any of the following values (0-2): 0—REAUTH_TYPE_DEFAULT 1—REAUTH_TYPE_LAST 2—REAUTH_TYPE_RERUN **Note** If you do not know the NAS IP address, you can enter the required values up to that point and the API will use these values in its search query. However, you must know the MAC address to perform this API call, but you can leave other parameters as null. This API call can only be executed on a Monitoring ISE node, which submits the requests to perform CoA remotely. The Administration ISE node is not involved or required to execute these CoA API calls.

Table 1-3 Cisco ISE Change of Authorization API Calls (continued)

API Call	Description and Example
Session Disconnect	
Disconnect	Sends a session disconnect command and port option type. https://<ISEhost>/ise/mnt/CoA/Disconnect/<serverhostname>/<macaddress>/<disconnecttype>/<nasipaddress>/<destinationipaddress> Port option type can be any of the following values (0-2): 0—DYNAMIC_AUTHZ_PORT_DEFAULT 1—DYNAMIC_AUTHZ_PORT_BOUNCE 2—DYNAMIC_AUTHZ_PORT_SHUTDOWN **Note** If you do not know the NAS IP address, enter the required values up to that point and the API will use these values in its search query. However, you must know the MAC address to perform this API call, but you can leave other parameters as null.

For details about Cisco ISE Change of Authorization API calls, see Chapter 4, "Using Change of Authorization REST APIs".

HTTP PUT API Calls

Similar to AuthStatus API call in Table 1-2, there is an HTTP PUT version of an API call that allows clients to retrieve account status. The Monitoring REST API supports both HTTP PUT and HTTP GET calls, with the examples in this guide documenting HTTP GET calls. HTTP PUT addresses the need for calls that require parameter inputs. The following schema file example is a request for account status:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="acctRequest" type="mnTRESTAcctRequest"/>

  <xs:complexType name="mnTRESTAcctRequest">
    <xs:complexContent>
      <xs:extension base="mnTRESTRequest">
        <xs:sequence>
          <xs:element name="duration" type="xs:string" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="mnTRESTRequest" abstract="true">
    <xs:sequence>
      <xs:element name="valueList">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="value" type="xs:string" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="searchCriteria" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

```
</xs:schema>
```

Using API Calls for Session Management

Cisco ISE session-management API calls use a Monitoring node to retrieve session-related information. The following sections describe each type of call as well as provide output schema file examples, procedures for issuing each call, a sample of the data returned, and how to remove stale sessions:

Session Counter API Calls

The following API calls let you quickly gather a current count of session-related information on a targeted Monitoring node in your Cisco ISE deployment:

- ActiveCount—counts active sessions.
- PostureCount—counts postured sessions.
- ProfilerCount—counts profiled sessions.

Using ActiveCount API Calls

You use an ActiveCount API call to retrieve a count of active sessions. This section contains the following sections:

ActiveCount API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="sessionCount" type="activeCount"/>
  <xs:complexType name="activeCount">
    <xs:sequence>
      <xs:element name="count" type="xs:int"/>
```

```
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Invoking the ActiveCount API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the ActiveCount API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/Session/<specific-api-call>):

```
https://acme123/ise/mnt/Session/ActiveCount
```

Note You must carefully enter each API call in the URL address field of a target node because these calls are case sensitive. The use of "mnt" in the API call convention represents the target Monitoring node.

Step 3 Press **Enter** to issue the API call.

ActiveCount API Call Data

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.
-
<sessionCount>
<count>5</count>
</sessionCount>
```

Using PostureCount API Call

You use a PostureCount API call to retrieve a count of active Posture sessions. This section contains the following sections:

PostureCount API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
  <xs:element name="sessionCount" type="postureCount"/>

  <xs:complexType name="postureCount">
    <xs:sequence>
      <xs:element name="count" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Invoking a PostureCount API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the PostureCount API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/Session/<specific-api-call>):

```
https://acme123/ise/mnt/Session/PostureCount
```

Note You must carefully enter each API call in the URL address field of a target node because these calls are case sensitive. The use of "mnt" in the API call convention represents the target Monitoring node.

Step 3 Press **Enter** to issue the API call.

PostureCount API Call Data

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<sessionCount>
<count>3</count>
</sessionCount>
```

Using ProfilerCount API Call

You use the ProfilerCount API call to retrieve a count of active Profiler sessions. This section contains the following sections:

ProfilerCount API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="sessionCount" type="profilerCount"/>

  <xs:complexType name="profilerCount">
    <xs:sequence>
      <xs:element name="count" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Invoking a ProfilerCount API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the ProfilerCount API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/Session/<specific-api-call>):

```
https://acme123/ise/mnt/Session/ProfilerCount
```

Note You must carefully enter each API call in the URL address field of a target node because these calls are case sensitive. The use of "mnt" in the API call convention represents the target Monitoring node.

Step 3 Press **Enter** to issue the API call.

ProfilerCount API Call Data

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<sessionCount>
<count>1</count>
</sessionCount>
```

Session List API Calls

The following session list API calls let you quickly gather session-related information such as the MAC address, the network access device (NAD) IP address, username, and session ID associated with a current active session on a target Monitoring node in a Cisco ISE deployment:

- ActiveList—lists active sessions
- AuthList—lists authenticated sessions

Using ActiveList API Calls

You use an ActiveList API call to list all active sessions. This section contains the following sections:

- ActiveList API Call Schema File, page 2-5
- Invoking an ActiveList API Call, page 2-6
- ActiveList API Call Data, page 2-6

Note In this release of Cisco ISE, the maximum number of active authenticated endpoint sessions that can be displayed is 100,000.

ActiveList API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="activeSessionList" type="simpleActiveSessionList"/>

<xs:complexType name="simpleActiveSessionList">
    <xs:sequence>
      <xs:element name="activeSession" type="simpleActiveSession" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="noOfActiveSession" type="xs:int" use="required"/>
  </xs:complexType>

  <xs:complexType name="simpleActiveSession">
    <xs:sequence>
      <xs:element name="user_name" type="xs:string" minOccurs="0"/>
      <xs:element name="calling_station_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_ip_address" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="audit_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="server" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Invoking an ActiveList API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the ActiveList API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/Session/<specific-api-call>):

```
https://acme123/ise/mnt/Session/ActiveList
```

Note You must carefully enter each API call in the URL address field of a target node because these calls are case sensitive. The use of "mnt" in the API call convention represents the target Monitoring node.

Step 3 Press **Enter** to issue the API call.

ActiveList API Call Data

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<activeSessionList noOfActiveSession="5">
-
<activeSession>
<calling_station_id>00:0C:29:FA:EF:0A</calling_station_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
<calling_station_id>70:5A:B6:68:F7:CC</calling_station_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
<user_name>tom_wolfe</user_name>
<calling_station_id>00:14:BF:5A:0C:03</calling_station_id>
<nas_ip_address>10.203.107.161</nas_ip_address>
<acct_session_id>00000032</acct_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
<user_name>graham_hancock</user_name>
<calling_station_id>00:50:56:8E:28:BD</calling_station_id>
<nas_ip_address>10.203.107.161</nas_ip_address>
<acct_session_id>0000002C</acct_session_id>
<audit_session_id>0ACB6BA10000002A165FD0C8</audit_session_id>
```

```
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
<user_name>ipepvpnuser</user_name>
<calling_station_id>172.23.130.89</calling_station_id>
<nas_ip_address>10.203.107.45</nas_ip_address>
<acct_session_id>A2000070</acct_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
</activeSessionList>
```

Using AuthList API Calls

You use an AuthList API call to retrieve a list of all active authenticated sessions. This section contains the following sections:

Note In this release of Cisco ISE, the maximum number of active authenticated endpoint sessions that can be displayed is 100,000.

AuthList API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="activeSessionList" type="simpleActiveSessionList"/>

<xs:complexType name="simpleActiveSessionList">
    <xs:sequence>
      <xs:element name="activeSession" type="simpleActiveSession" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="noOfActiveSession" type="xs:int" use="required"/>
  </xs:complexType>

  <xs:complexType name="simpleActiveSession">
    <xs:sequence>
      <xs:element name="user_name" type="xs:string" minOccurs="0"/>
      <xs:element name="calling_station_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_ip_address" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="audit_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="server" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Invoking an AuthList API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the AuthList API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/Session/<specific-api-call>):

Note The first of the following two examples uses a defined start time and null parameter, which displays a list of the currently active sessions that were authenticated after the specified start time. The second example uses the null/null parameter that displays a list of all currently active authenticated sessions. See AuthList API Call Data, page 2-8, which displays samples of the four parameter setting types for this API call.

```
https://acme123/ise/mnt/Session/AuthList/2010-12-14 15:33:15/null

https://acme123/ise/mnt/Session/AuthList/null/null
```

Note You must carefully enter each API call in the URL address field of a target node because these calls are case sensitive. The use of "mnt" in the API call convention represents the targeted Monitoring node.

Step 3 Press **Enter** to issue the API call.

AuthList API Call Data

Using the null/null Option

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<activeSessionList noOfActiveSession="3">
-
<activeSession>
<user_name>ipepwlcuser</user_name>
<calling_station_id>00:26:82:7B:D2:51</calling_station_id>
<nas_ip_address>10.203.107.10</nas_ip_address>
<audit_session_id>0acb6b0c000000174D07F487</audit_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
<user_name>tom_wolfe</user_name>
<calling_station_id>00:50:56:8E:28:BD</calling_station_id>
```

```
<nas_ip_address>10.203.107.161</nas_ip_address>
<acct_session_id>00000035</acct_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
<user_name>graham_hancock</user_name>
<calling_station_id>00:14:BF:5A:0C:03</calling_station_id>
<nas_ip_address>10.203.107.161</nas_ip_address>
<acct_session_id>00000033</acct_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
</activeSessionList>
```

Using the endtime/null Option

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<activeSessionList noOfActiveSession="3">
-
<activeSession>
<user_name>ipepwlcuser</user_name>
<calling_station_id>00:26:82:7B:D2:51</calling_station_id>
<nas_ip_address>10.203.107.10</nas_ip_address>
<audit_session_id>0acb6b0c0000001F4D08085A</audit_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
<user_name>hunter_thompson</user_name>
<calling_station_id>00:50:56:8E:28:BD</calling_station_id>
<nas_ip_address>10.203.107.161</nas_ip_address>
<acct_session_id>00000035</acct_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
<user_name>bob_ludlum</user_name>
<calling_station_id>00:14:BF:5A:0C:03</calling_station_id>
<nas_ip_address>10.203.107.161</nas_ip_address>
<acct_session_id>00000033</acct_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
</activeSessionList>
```

Using the null/starttime Option

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<activeSessionList noOfActiveSession="3">
-
<activeSession>
<user_name>ipepwlcuser</user_name>
<calling_station_id>00:26:82:7B:D2:51</calling_station_id>
<nas_ip_address>10.203.107.10</nas_ip_address>
<audit_session_id>0acb6b0c0000001F4D08085A</audit_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
```

```
<user_name>bob_ludlum</user_name>
<calling_station_id>00:50:56:8E:28:BD</calling_station_id>
<nas_ip_address>10.203.107.161</nas_ip_address>
<acct_session_id>00000035</acct_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
<user_name>tom_wolfe</user_name>
<calling_station_id>00:14:BF:5A:0C:03</calling_station_id>
<nas_ip_address>10.203.107.161</nas_ip_address>
<acct_session_id>00000033</acct_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
</activeSessionList>
```

Using the starttime/endtime Option

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.
-
<activeSessionList noOfActiveSession="3">
-
<activeSession>
<user_name>ipepwlcuser</user_name>
<calling_station_id>00:26:82:7B:D2:51</calling_station_id>
<nas_ip_address>10.203.107.10</nas_ip_address>
<audit_session_id>0acb6b0c0000001F4D08085A</audit_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
<user_name>graham_hancock</user_name>
<calling_station_id>00:50:56:8E:28:BD</calling_station_id>
<nas_ip_address>10.203.107.161</nas_ip_address>
<acct_session_id>00000035</acct_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
-
<activeSession>
<user_name>hunter_thompson</user_name>
<calling_station_id>00:14:BF:5A:0C:03</calling_station_id>
<nas_ip_address>10.203.107.161</nas_ip_address>
<acct_session_id>00000033</acct_session_id>
<server>HAREESH-R6-1-PDP2</server>
</activeSession>
</activeSessionList>
```

Session Attribute API Calls

The following session attribute API calls let you quickly search the latest session for key information, such as the following:

- MAC address session search using a MACAddress API call
- User name session search using a UserName API call
- NAS IP address session search using an IPAddress API call

Using MACAddress API Calls

You use a MACAddress API call to retrieve a specified MAC address from an active session on a targeted monitoring node. This section contains the following sections:

MACAddress API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="sessionParameters" type="restsdStatus"/>

  <xs:complexType name="restsdStatus">
    <xs:sequence>
      <xs:element name="passed" type="xs:anyType" minOccurs="0"/>
      <xs:element name="failed" type="xs:anyType" minOccurs="0"/>
      <xs:element name="user_name" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_ip_address" type="xs:string" minOccurs="0"/>
      <xs:element name="failure_reason" type="xs:string" minOccurs="0"/>
      <xs:element name="calling_station_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_port" type="xs:string" minOccurs="0"/>
      <xs:element name="identity_group" type="xs:string" minOccurs="0"/>
      <xs:element name="network_device_name" type="xs:string" minOccurs="0"/>
      <xs:element name="acs_server" type="xs:string" minOccurs="0"/>
      <xs:element name="authen_protocol" type="xs:string" minOccurs="0"/>
      <xs:element name="framed_ip_address" type="xs:string" minOccurs="0"/>
      <xs:element name="network_device_groups" type="xs:string" minOccurs="0"/>
      <xs:element name="access_service" type="xs:string" minOccurs="0"/>
      <xs:element name="auth_acs_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="authentication_method" type="xs:string" minOccurs="0"/>
      <xs:element name="execution_steps" type="xs:string" minOccurs="0"/>
      <xs:element name="radius_response" type="xs:string" minOccurs="0"/>
      <xs:element name="audit_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_identifier" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_port_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_policy_compliance" type="xs:string" minOccurs="0"/>
      <xs:element name="auth_id" type="xs:long" minOccurs="0"/>
      <xs:element name="auth_acsview_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="message_code" type="xs:string" minOccurs="0"/>
      <xs:element name="acs_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="service_selection_policy" type="xs:string" minOccurs="0"/>
      <xs:element name="authorization_policy" type="xs:string" minOccurs="0"/>
      <xs:element name="identity_store" type="xs:string" minOccurs="0"/>
      <xs:element name="response" type="xs:string" minOccurs="0"/>
      <xs:element name="service_type" type="xs:string" minOccurs="0"/>
      <xs:element name="cts_security_group" type="xs:string" minOccurs="0"/>
      <xs:element name="use_case" type="xs:string" minOccurs="0"/>
      <xs:element name="cisco_av_pair" type="xs:string" minOccurs="0"/>
      <xs:element name="ad_domain" type="xs:string" minOccurs="0"/>
      <xs:element name="acs_username" type="xs:string" minOccurs="0"/>
      <xs:element name="radius_username" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_role" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_username" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_posture_token" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_radius_is_user_auth" type="xs:string" minOccurs="0"/>
      <xs:element name="selected_posture_server" type="xs:string" minOccurs="0"/>
```

```
      <xs:element name="selected_identity_store" type="xs:string" minOccurs="0"/>
      <xs:element name="authentication_identity_store" type="xs:string" minOccurs="0"/>
      <xs:element name="azn_exp_pol_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="ext_pol_server_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="grp_mapping_pol_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="identity_policy_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_port_type" type="xs:string" minOccurs="0"/>
      <xs:element name="query_identity_stores" type="xs:string" minOccurs="0"/>
      <xs:element name="selected_azn_profiles" type="xs:string" minOccurs="0"/>
      <xs:element name="sel_exp_azn_profiles" type="xs:string" minOccurs="0"/>
      <xs:element name="selected_query_identity_stores" type="xs:string" minOccurs="0"/>
      <xs:element name="eap_tunnel" type="xs:string" minOccurs="0"/>
      <xs:element name="tunnel_details" type="xs:string" minOccurs="0"/>
      <xs:element name="cisco_h323_attributes" type="xs:string" minOccurs="0"/>
      <xs:element name="cisco_ssg_attributes" type="xs:string" minOccurs="0"/>
      <xs:element name="other_attributes" type="xs:string" minOccurs="0"/>
      <xs:element name="response_time" type="xs:long" minOccurs="0"/>
      <xs:element name="nad_failure" type="xs:anyType" minOccurs="0"/>
      <xs:element name="destination_ip_address" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_id" type="xs:long" minOccurs="0"/>
      <xs:element name="acct_acs_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="acct_acsview_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="acct_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_status_type" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_session_time" type="xs:long" minOccurs="0"/>
      <xs:element name="acct_input_octets" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_output_octets" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_input_packets" type="xs:long" minOccurs="0"/>
      <xs:element name="acct_output_packets" type="xs:long" minOccurs="0"/>
      <xs:element name="acct_class" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_terminate_cause" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_multi_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_authentic" type="xs:string" minOccurs="0"/>
      <xs:element name="termination_action" type="xs:string" minOccurs="0"/>
      <xs:element name="session_timeout" type="xs:string" minOccurs="0"/>
      <xs:element name="idle_timeout" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_interim_interval" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_delay_time" type="xs:string" minOccurs="0"/>
      <xs:element name="event_timestamp" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_tunnel_connection" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_tunnel_packet_lost" type="xs:string" minOccurs="0"/>
      <xs:element name="security_group" type="xs:string" minOccurs="0"/>
      <xs:element name="cisco_h323_setup_time" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="cisco_h323_connect_time" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="cisco_h323_disconnect_time" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="framed_protocol" type="xs:string" minOccurs="0"/>
      <xs:element name="started" type="xs:anyType" minOccurs="0"/>
      <xs:element name="stopped" type="xs:anyType" minOccurs="0"/>
      <xs:element name="ckpt_id" type="xs:long" minOccurs="0"/>
      <xs:element name="type" type="xs:long" minOccurs="0"/>
      <xs:element name="nad_acsview_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="vlan" type="xs:string" minOccurs="0"/>
      <xs:element name="dacl" type="xs:string" minOccurs="0"/>
      <xs:element name="authentication_type" type="xs:string" minOccurs="0"/>
      <xs:element name="interface_name" type="xs:string" minOccurs="0"/>
      <xs:element name="reason" type="xs:string" minOccurs="0"/>
      <xs:element name="endpoint_policy" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Invoking a MACAddress API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the MACAddress API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/Session/<specific-api-call>/<macaddress>):

```
https://acme123/ise/mnt/Session/MACAddress/0A:0B:0C:0D:0E:0F
```

Note Make sure that you specify the MAC address using the XX:XX:XX:XX:XX:XX format.

Note You must carefully enter each API call in the URL address field of a target node because these calls are case sensitive. The use of "mnt" in the API call convention represents the target Monitoring node.

Step 3 Press **Enter** to issue the API call.

Note This API call returns only the session data that is created during the last 5 days.

MACAddress API Call Data

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<sessionParameters>
<passed xsi:type="xs:boolean">true</passed>
<failed xsi:type="xs:boolean">false</failed>
<user_name>hunter_thompson</user_name>
<nas_ip_address>10.203.107.161</nas_ip_address>
<calling_station_id>00:14:BF:5A:0C:03</calling_station_id>
<nas_port>50115</nas_port>
<identity_group>Profiled</identity_group>
<network_device_name>Core-Switch</network_device_name>
<acs_server>HAREESH-R6-1-PDP2</acs_server>
<authen_protocol>Lookup</authen_protocol>
-
<network_device_groups>
Device Type#All Device Types,Location#All Locations
</network_device_groups>
<access_service>RADIUS</access_service>
<auth_acs_timestamp>2010-12-15T02:11:12.359Z</auth_acs_timestamp>
```

<authentication_method>mab</authentication_method>
-
<execution_steps>
11001,11017,11027,15008,15048,15004,15041,15004,15013,24209,24211,22037,15036,15048,15048,
15004,15016,11022,11002
</execution_steps>
<audit_session_id>0ACB6BA1000000351BBFBF8B</audit_session_id>
<nas_port_id>GigabitEthernet1/0/15</nas_port_id>
<nac_policy_compliance>Pending</nac_policy_compliance>
<auth_id>1291240762077361</auth_id>
<auth_acsview_timestamp>2010-12-15T02:11:12.360Z</auth_acsview_timestamp>
<message_code>5200</message_code>
<acs_session_id>HAREESH-R6-1-PDP2/81148292/681</acs_session_id>
<service_selection_policy>MAB</service_selection_policy>
<identity_store>Internal Hosts</identity_store>
-
<response>
{UserName=00-14-BF-5A-0C-03; User-Name=00-14-BF-5A-0C-03;
State=ReauthSession:0ACB6BA1000000351BBFBF8B;
Class=CACS:0ACB6BA1000000351BBFBF8B:HAREESH-R6-1-PDP2/81148292/681;
Termination-Action=RADIUS-Request; cisco-av-pair=url-redirect-acl=ACL-WEBAUTH-REDIRECT;
cisco-av-pair=url-redirect=https://HAREESH-R6-1-PDP2.cisco.com:8443/guestportal/gateway?se
ssionId=0ACB6BA1000000351BBFBF8B&action=cwa;
cisco-av-pair=ACS:CiscoSecure-Defined-ACL=#ACSACL#-IP-ACL-DENY-4ced8390; }
</response>
<service_type>Call Check</service_type>
<use_case>Host Lookup</use_case>
<cisco_av_pair>audit-session-id=0ACB6BA1000000351BBFBF8B</cisco_av_pair>
<acs_username>00:14:BF:5A:0C:03</acs_username>
<radius_username>00:14:BF:5A:0C:03</radius_username>
<selected_identity_store>Internal Hosts</selected_identity_store>
<authentication_identity_store>Internal Hosts</authentication_identity_store>
<identity_policy_matched_rule>Default</identity_policy_matched_rule>
<nas_port_type>Ethernet</nas_port_type>
<selected_azn_profiles>CWA</selected_azn_profiles>
-
<other_attributes>
ConfigVersionId=44,DestinationIPAddress=10.203.107.162,DestinationPort=1812,Protocol=Radiu
s,Framed-MTU=1500,EAP-Key-Name=,CPMSessionID=0ACB6BA1000000351BBFBF8B,CPMSessionID=0ACB6BA
1000000351BBFBF8B,EndPointMACAddress=00-14-BF-5A-0C-03,HostIdentityGroup=Endpoint Identity
Groups:Profiled,Device Type=Device Type#All Device Types,Location=Location#All
Locations,Model Name=Unknown,Software Version=Unknown,Device IP
Address=10.203.107.161,Called-Station-ID=04:FE:7F:7F:C0:8F
</other_attributes>
<response_time>77</response_time>
<acct_id>1291240762077386</acct_id>
<acct_acs_timestamp>2010-12-15T02:12:30.779Z</acct_acs_timestamp>
<acct_acsview_timestamp>2010-12-15T02:12:30.780Z</acct_acsview_timestamp>
<acct_session_id>00000038</acct_session_id>
<acct_status_type>Interim-Update</acct_status_type>
<acct_session_time>78</acct_session_time>
<acct_input_octets>13742</acct_input_octets>
<acct_output_octets>6277</acct_output_octets>
<acct_input_packets>108</acct_input_packets>
<acct_output_packets>66</acct_output_packets>
-
<acct_class>
CACS:0ACB6BA1000000351BBFBF8B:HAREESH-R6-1-PDP2/81148292/681
</acct_class>
<acct_delay_time>0</acct_delay_time>
<started xsi:type="xs:boolean">false</started>
<stopped xsi:type="xs:boolean">false</stopped>
</sessionParameters>

Using UserName API Calls

You use a UserName API call to retrieve a specified username from an active session. This section contains the following sections:

UserName API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="sessionParameters" type="restsdStatus"/>

  <xs:complexType name="restsdStatus">
    <xs:sequence>
      <xs:element name="passed" type="xs:anyType" minOccurs="0"/>
      <xs:element name="failed" type="xs:anyType" minOccurs="0"/>
      <xs:element name="user_name" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_ip_address" type="xs:string" minOccurs="0"/>
      <xs:element name="failure_reason" type="xs:string" minOccurs="0"/>
      <xs:element name="calling_station_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_port" type="xs:string" minOccurs="0"/>
      <xs:element name="identity_group" type="xs:string" minOccurs="0"/>
      <xs:element name="network_device_name" type="xs:string" minOccurs="0"/>
      <xs:element name="acs_server" type="xs:string" minOccurs="0"/>
      <xs:element name="authen_protocol" type="xs:string" minOccurs="0"/>
      <xs:element name="framed_ip_address" type="xs:string" minOccurs="0"/>
      <xs:element name="network_device_groups" type="xs:string" minOccurs="0"/>
      <xs:element name="access_service" type="xs:string" minOccurs="0"/>
      <xs:element name="auth_acs_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="authentication_method" type="xs:string" minOccurs="0"/>
      <xs:element name="execution_steps" type="xs:string" minOccurs="0"/>
      <xs:element name="radius_response" type="xs:string" minOccurs="0"/>
      <xs:element name="audit_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_identifier" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_port_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_policy_compliance" type="xs:string" minOccurs="0"/>
      <xs:element name="auth_id" type="xs:long" minOccurs="0"/>
      <xs:element name="auth_acsview_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="message_code" type="xs:string" minOccurs="0"/>
      <xs:element name="acs_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="service_selection_policy" type="xs:string" minOccurs="0"/>
      <xs:element name="authorization_policy" type="xs:string" minOccurs="0"/>
      <xs:element name="identity_store" type="xs:string" minOccurs="0"/>
      <xs:element name="response" type="xs:string" minOccurs="0"/>
      <xs:element name="service_type" type="xs:string" minOccurs="0"/>
      <xs:element name="cts_security_group" type="xs:string" minOccurs="0"/>
      <xs:element name="use_case" type="xs:string" minOccurs="0"/>
      <xs:element name="cisco_av_pair" type="xs:string" minOccurs="0"/>
      <xs:element name="ad_domain" type="xs:string" minOccurs="0"/>
      <xs:element name="acs_username" type="xs:string" minOccurs="0"/>
      <xs:element name="radius_username" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_role" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_username" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_posture_token" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_radius_is_user_auth" type="xs:string" minOccurs="0"/>
      <xs:element name="selected_posture_server" type="xs:string" minOccurs="0"/>
```

```
      <xs:element name="selected_identity_store" type="xs:string" minOccurs="0"/>
      <xs:element name="authentication_identity_store" type="xs:string" minOccurs="0"/>
      <xs:element name="azn_exp_pol_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="ext_pol_server_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="grp_mapping_pol_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="identity_policy_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_port_type" type="xs:string" minOccurs="0"/>
      <xs:element name="query_identity_stores" type="xs:string" minOccurs="0"/>
      <xs:element name="selected_azn_profiles" type="xs:string" minOccurs="0"/>
      <xs:element name="sel_exp_azn_profiles" type="xs:string" minOccurs="0"/>
      <xs:element name="selected_query_identity_stores" type="xs:string" minOccurs="0"/>
      <xs:element name="eap_tunnel" type="xs:string" minOccurs="0"/>
      <xs:element name="tunnel_details" type="xs:string" minOccurs="0"/>
      <xs:element name="cisco_h323_attributes" type="xs:string" minOccurs="0"/>
      <xs:element name="cisco_ssg_attributes" type="xs:string" minOccurs="0"/>
      <xs:element name="other_attributes" type="xs:string" minOccurs="0"/>
      <xs:element name="response_time" type="xs:long" minOccurs="0"/>
      <xs:element name="nad_failure" type="xs:anyType" minOccurs="0"/>
      <xs:element name="destination_ip_address" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_id" type="xs:long" minOccurs="0"/>
      <xs:element name="acct_acs_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="acct_acsview_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="acct_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_status_type" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_session_time" type="xs:long" minOccurs="0"/>
      <xs:element name="acct_input_octets" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_output_octets" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_input_packets" type="xs:long" minOccurs="0"/>
      <xs:element name="acct_output_packets" type="xs:long" minOccurs="0"/>
      <xs:element name="acct_class" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_terminate_cause" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_multi_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_authentic" type="xs:string" minOccurs="0"/>
      <xs:element name="termination_action" type="xs:string" minOccurs="0"/>
      <xs:element name="session_timeout" type="xs:string" minOccurs="0"/>
      <xs:element name="idle_timeout" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_interim_interval" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_delay_time" type="xs:string" minOccurs="0"/>
      <xs:element name="event_timestamp" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_tunnel_connection" type="xs:string" minOccurs="0"/>
      <xs:element name="acct_tunnel_packet_lost" type="xs:string" minOccurs="0"/>
      <xs:element name="security_group" type="xs:string" minOccurs="0"/>
      <xs:element name="cisco_h323_setup_time" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="cisco_h323_connect_time" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="cisco_h323_disconnect_time" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="framed_protocol" type="xs:string" minOccurs="0"/>
      <xs:element name="started" type="xs:anyType" minOccurs="0"/>
      <xs:element name="stopped" type="xs:anyType" minOccurs="0"/>
      <xs:element name="ckpt_id" type="xs:long" minOccurs="0"/>
      <xs:element name="type" type="xs:long" minOccurs="0"/>
      <xs:element name="nad_acsview_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="vlan" type="xs:string" minOccurs="0"/>
      <xs:element name="dacl" type="xs:string" minOccurs="0"/>
      <xs:element name="authentication_type" type="xs:string" minOccurs="0"/>
      <xs:element name="interface_name" type="xs:string" minOccurs="0"/>
      <xs:element name="reason" type="xs:string" minOccurs="0"/>
      <xs:element name="endpoint_policy" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Invoking a UserName API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the UserName API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/Session/<specific-api-call>/<username>):

```
https://acme123/ise/mnt/Session/UserName/graham_hancock
```

Note You must carefully enter each API call in the URL address field of a target node because these calls are case sensitive. The use of "mnt" in the API call convention represents the target Monitoring node.

Step 3 Press **Enter** to issue the API call.

UserName API Call Data

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<sessionParameters>
<passed xsi:type="xs:boolean">true</passed>
<failed xsi:type="xs:boolean">false</failed>
<user_name>graham_hancock</user_name>
<nas_ip_address>10.203.107.161</nas_ip_address>
<calling_station_id>00:14:BF:5A:0C:03</calling_station_id>
<nas_port>50115</nas_port>
<identity_group>Profiled</identity_group>
<network_device_name>Core-Switch</network_device_name>
<acs_server>HAREESH-R6-1-PDP2</acs_server>
<authen_protocol>Lookup</authen_protocol>
-
<network_device_groups>
Device Type#All Device Types,Location#All Locations
</network_device_groups>
<access_service>RADIUS</access_service>
<auth_acs_timestamp>2010-12-15T02:11:12.359Z</auth_acs_timestamp>
<authentication_method>mab</authentication_method>
-
<execution_steps>
11001,11017,11027,15008,15048,15004,15041,15004,15013,24209,24211,22037,15036,15048,15048,
15004,15016,11022,11002
</execution_steps>
<audit_session_id>0ACB6BA1000000351BBFBF8B</audit_session_id>
<nas_port_id>GigabitEthernet1/0/15</nas_port_id>
<nac_policy_compliance>Pending</nac_policy_compliance>
```

```
<auth_id>1291240762077361</auth_id>
<auth_acsview_timestamp>2010-12-15T02:11:12.360Z</auth_acsview_timestamp>
<message_code>5200</message_code>
<acs_session_id>HAREESH-R6-1-PDP2/81148292/681</acs_session_id>
<service_selection_policy>MAB</service_selection_policy>
<identity_store>Internal Hosts</identity_store>
-
<response>
{UserName=graham_hancock; User-Name=graham_hancock;
State=ReauthSession:0ACB6BA1000000351BBFBF8B;
Class=CACS:0ACB6BA1000000351BBFBF8B:HAREESH-R6-1-PDP2/81148292/681;
Termination-Action=RADIUS-Request; cisco-av-pair=url-redirect-acl=ACL-WEBAUTH-REDIRECT;
cisco-av-pair=url-redirect=https://HAREESH-R6-1-PDP2.cisco.com:8443/guestportal/gateway?se
ssionId=0ACB6BA1000000351BBFBF8B&action=cwa;
cisco-av-pair=ACS:CiscoSecure-Defined-ACL=#ACSACL#-IP-ACL-DENY-4ced8390; }
</response>
<service_type>Call Check</service_type>
<use_case>Host Lookup</use_case>
<cisco_av_pair>audit-session-id=0ACB6BA1000000351BBFBF8B</cisco_av_pair>
<acs_username>graham_hancock</acs_username>
<radius_username>00:14:BF:5A:0C:03</radius_username>
<selected_identity_store>Internal Hosts</selected_identity_store>
<authentication_identity_store>Internal Hosts</authentication_identity_store>
<identity_policy_matched_rule>Default</identity_policy_matched_rule>
<nas_port_type>Ethernet</nas_port_type>
<selected_azn_profiles>CWA</selected_azn_profiles>
-
<other_attributes>
ConfigVersionId=44,DestinationIPAddress=10.203.107.162,DestinationPort=1812,Protocol=Radiu
s,Framed-MTU=1500,EAP-Key-Name=,CPMSessionID=0ACB6BA1000000351BBFBF8B,CPMSessionID=0ACB6BA
1000000351BBFBF8B,EndPointMACAddress=00-14-BF-5A-0C-03,HostIdentityGroup=Endpoint Identity
Groups:Profiled,Device Type=Device Type#All Device Types,Location=Location#All
Locations,Model Name=Unknown,Software Version=Unknown,Device IP
Address=10.203.107.161,Called-Station-ID=04:FE:7F:7F:C0:8F
</other_attributes>
<response_time>77</response_time>
<acct_id>1291240762077386</acct_id>
<acct_acs_timestamp>2010-12-15T02:12:30.779Z</acct_acs_timestamp>
<acct_acsview_timestamp>2010-12-15T02:12:30.780Z</acct_acsview_timestamp>
<acct_session_id>00000038</acct_session_id>
<acct_status_type>Interim-Update</acct_status_type>
<acct_session_time>78</acct_session_time>
<acct_input_octets>13742</acct_input_octets>
<acct_output_octets>6277</acct_output_octets>
<acct_input_packets>108</acct_input_packets>
<acct_output_packets>66</acct_output_packets>
-
<acct_class>
CACS:0ACB6BA1000000351BBFBF8B:HAREESH-R6-1-PDP2/81148292/681
</acct_class>
<acct_delay_time>0</acct_delay_time>
<started xsi:type="xs:boolean">false</started>
<stopped xsi:type="xs:boolean">false</stopped>
</sessionParameters>
```

Using IPAddress API Calls

You use an IPAddress API call to retrieve data for a specified network access server (NAS) IP address from a current session. This section contains the following sections:

IPAddress API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="sessionParameters" type="restsdStatus"/>

  <xs:complexType name="restsdStatus">
    <xs:sequence>
      <xs:element name="passed" type="xs:anyType" minOccurs="0"/>
      <xs:element name="failed" type="xs:anyType" minOccurs="0"/>
      <xs:element name="user_name" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_ip_address" type="xs:string" minOccurs="0"/>
      <xs:element name="failure_reason" type="xs:string" minOccurs="0"/>
      <xs:element name="calling_station_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_port" type="xs:string" minOccurs="0"/>
      <xs:element name="identity_group" type="xs:string" minOccurs="0"/>
      <xs:element name="network_device_name" type="xs:string" minOccurs="0"/>
      <xs:element name="acs_server" type="xs:string" minOccurs="0"/>
      <xs:element name="authen_protocol" type="xs:string" minOccurs="0"/>
      <xs:element name="framed_ip_address" type="xs:string" minOccurs="0"/>
      <xs:element name="network_device_groups" type="xs:string" minOccurs="0"/>
      <xs:element name="access_service" type="xs:string" minOccurs="0"/>
      <xs:element name="auth_acs_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="authentication_method" type="xs:string" minOccurs="0"/>
      <xs:element name="execution_steps" type="xs:string" minOccurs="0"/>
      <xs:element name="radius_response" type="xs:string" minOccurs="0"/>
      <xs:element name="audit_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_identifier" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_port_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_policy_compliance" type="xs:string" minOccurs="0"/>
      <xs:element name="auth_id" type="xs:long" minOccurs="0"/>
      <xs:element name="auth_acsview_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="message_code" type="xs:string" minOccurs="0"/>
      <xs:element name="acs_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="service_selection_policy" type="xs:string" minOccurs="0"/>
      <xs:element name="authorization_policy" type="xs:string" minOccurs="0"/>
      <xs:element name="identity_store" type="xs:string" minOccurs="0"/>
      <xs:element name="response" type="xs:string" minOccurs="0"/>
      <xs:element name="service_type" type="xs:string" minOccurs="0"/>
      <xs:element name="cts_security_group" type="xs:string" minOccurs="0"/>
      <xs:element name="use_case" type="xs:string" minOccurs="0"/>
      <xs:element name="cisco_av_pair" type="xs:string" minOccurs="0"/>
      <xs:element name="ad_domain" type="xs:string" minOccurs="0"/>
      <xs:element name="acs_username" type="xs:string" minOccurs="0"/>
      <xs:element name="radius_username" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_role" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_username" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_posture_token" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_radius_is_user_auth" type="xs:string" minOccurs="0"/>
      <xs:element name="selected_posture_server" type="xs:string" minOccurs="0"/>
      <xs:element name="selected_identity_store" type="xs:string" minOccurs="0"/>
      <xs:element name="authentication_identity_store" type="xs:string" minOccurs="0"/>
      <xs:element name="azn_exp_pol_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="ext_pol_server_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="grp_mapping_pol_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="identity_policy_matched_rule" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_port_type" type="xs:string" minOccurs="0"/>
```

```
        <xs:element name="query_identity_stores" type="xs:string" minOccurs="0"/>
        <xs:element name="selected_azn_profiles" type="xs:string" minOccurs="0"/>
        <xs:element name="sel_exp_azn_profiles" type="xs:string" minOccurs="0"/>
        <xs:element name="selected_query_identity_stores" type="xs:string" minOccurs="0"/>
        <xs:element name="eap_tunnel" type="xs:string" minOccurs="0"/>
        <xs:element name="tunnel_details" type="xs:string" minOccurs="0"/>
        <xs:element name="cisco_h323_attributes" type="xs:string" minOccurs="0"/>
        <xs:element name="cisco_ssg_attributes" type="xs:string" minOccurs="0"/>
        <xs:element name="other_attributes" type="xs:string" minOccurs="0"/>
        <xs:element name="response_time" type="xs:long" minOccurs="0"/>
        <xs:element name="nad_failure" type="xs:anyType" minOccurs="0"/>
        <xs:element name="destination_ip_address" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_id" type="xs:long" minOccurs="0"/>
        <xs:element name="acct_acs_timestamp" type="xs:dateTime" minOccurs="0"/>
        <xs:element name="acct_acsview_timestamp" type="xs:dateTime" minOccurs="0"/>
        <xs:element name="acct_session_id" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_status_type" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_session_time" type="xs:long" minOccurs="0"/>
        <xs:element name="acct_input_octets" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_output_octets" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_input_packets" type="xs:long" minOccurs="0"/>
        <xs:element name="acct_output_packets" type="xs:long" minOccurs="0"/>
        <xs:element name="acct_class" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_terminate_cause" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_multi_session_id" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_authentic" type="xs:string" minOccurs="0"/>
        <xs:element name="termination_action" type="xs:string" minOccurs="0"/>
        <xs:element name="session_timeout" type="xs:string" minOccurs="0"/>
        <xs:element name="idle_timeout" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_interim_interval" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_delay_time" type="xs:string" minOccurs="0"/>
        <xs:element name="event_timestamp" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_tunnel_connection" type="xs:string" minOccurs="0"/>
        <xs:element name="acct_tunnel_packet_lost" type="xs:string" minOccurs="0"/>
        <xs:element name="security_group" type="xs:string" minOccurs="0"/>
        <xs:element name="cisco_h323_setup_time" type="xs:dateTime" minOccurs="0"/>
        <xs:element name="cisco_h323_connect_time" type="xs:dateTime" minOccurs="0"/>
        <xs:element name="cisco_h323_disconnect_time" type="xs:dateTime" minOccurs="0"/>
        <xs:element name="framed_protocol" type="xs:string" minOccurs="0"/>
        <xs:element name="started" type="xs:anyType" minOccurs="0"/>
        <xs:element name="stopped" type="xs:anyType" minOccurs="0"/>
        <xs:element name="ckpt_id" type="xs:long" minOccurs="0"/>
        <xs:element name="type" type="xs:long" minOccurs="0"/>
        <xs:element name="nad_acsview_timestamp" type="xs:dateTime" minOccurs="0"/>
        <xs:element name="vlan" type="xs:string" minOccurs="0"/>
        <xs:element name="dacl" type="xs:string" minOccurs="0"/>
        <xs:element name="authentication_type" type="xs:string" minOccurs="0"/>
        <xs:element name="interface_name" type="xs:string" minOccurs="0"/>
        <xs:element name="reason" type="xs:string" minOccurs="0"/>
        <xs:element name="endpoint_policy" type="xs:string" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Invoking an IPAddress API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the IPAddress API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/Session/<specific-api-call>/<nasipaddress>):

```
https://acme123/ise/mnt/Session/IPAddress/10.10.10.10
```

Note Make sure that you specify the NAS IP address using the xxx.xxx.xxx.xxx format.

Note You must carefully enter each API call in the URL address field of a target node because these calls are case sensitive. The use of "mnt" in the API call convention represents the target Monitoring node.

Step 3 Press **Enter** to issue the API call.

Note This API call returns only the session data that is created during the last 5 days.

IPAddress API Call Data

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.
-
<sessionParameters>
<passed xsi:type="xs:boolean">true</passed>
<failed xsi:type="xs:boolean">false</failed>
<user_name>ipepvpnuser</user_name>
<nas_ip_address>10.10.10.10</nas_ip_address>
<calling_station_id>172.23.130.90</calling_station_id>
<nas_port>1015</nas_port>
<identity_group>iPEP-VPN-Group</identity_group>
<network_device_name>iPEP-HA-Routed</network_device_name>
<acs_server>HAREESH-R6-1-PDP2</acs_server>
<authen_protocol>PAP_ASCII</authen_protocol>
-
<network_device_groups>
Device Type#All Device Types,Location#All Locations
</network_device_groups>
<access_service>RADIUS</access_service>
<auth_acs_timestamp>2010-12-15T19:57:29.885Z</auth_acs_timestamp>
<authentication_method>PAP_ASCII</authentication_method>
-
<execution_steps>
11001,11017,15008,15048,15048,15004,15041,15004,15013,24210,24212,22037,15036,15048,15048,
15004,15016,11002
</execution_steps>
<audit_session_id>0acb6be4000000044D091DA9</audit_session_id>
<nac_policy_compliance>NotApplicable</nac_policy_compliance>
<auth_id>1291240762083580</auth_id>
```

```
<auth_acsview_timestamp>2010-12-15T19:57:29.887Z</auth_acsview_timestamp>
<message_code>5200</message_code>
<acs_session_id>HAREESH-R6-1-PDP2/81148292/693</acs_session_id>
<service_selection_policy>iPEP-VPN</service_selection_policy>
<identity_store>Internal Users</identity_store>
-
<response>
{User-Name=ipepvpnuser; State=ReauthSession:0acb6be4000000044D091DA9;
Class=CACS:0acb6be4000000044D091DA9:HAREESH-R6-1-PDP2/81148292/693;
Termination-Action=RADIUS-Request; }
</response>
<service_type>Framed</service_type>
-
<cisco_av_pair>
audit-session-id=0acb6be4000000044D091DA9,ipep-proxy=true
</cisco_av_pair>
<acs_username>ipepvpnuser</acs_username>
<radius_username>ipepvpnuser</radius_username>
<selected_identity_store>Internal Users</selected_identity_store>
<authentication_identity_store>Internal Users</authentication_identity_store>
<identity_policy_matched_rule>Default</identity_policy_matched_rule>
<nas_port_type>Virtual</nas_port_type>
<selected_azn_profiles>iPEP-Unknown-Auth-Profile</selected_azn_profiles>
<tunnel_details>Tunnel-Client-Endpoint=(tag=0) 172.23.130.90</tunnel_details>
-
<other_attributes>
ConfigVersionId=44,DestinationIPAddress=10.203.107.162,DestinationPort=1812,Protocol=Radiu
s,Framed-Protocol=PPP,Proxy-State=Cisco Secure
ACS9e733142-070a-11e0-c000-000000000000-2906094480-3222,CPMSessionID=0acb6be4000000044D091
DA9,CPMSessionID=0acb6be4000000044D091DA9,Device Type=Device Type#All Device
Types,Location=Location#All Locations,Model Name=Unknown,Software Version=Unknown,Device
IP Address=10.203.107.228,Called-Station-ID=172.23.130.94
</other_attributes>
<response_time>20</response_time>
<acct_id>1291240762083582</acct_id>
<acct_acs_timestamp>2010-12-15T19:57:30.281Z</acct_acs_timestamp>
<acct_acsview_timestamp>2010-12-15T19:57:30.283Z</acct_acsview_timestamp>
<acct_session_id>F1800007</acct_session_id>
<acct_status_type>Start</acct_status_type>
-
<acct_class>
CACS:0acb6be4000000044D091DA9:HAREESH-R6-1-PDP2/81148292/693
</acct_class>
<acct_delay_time>0</acct_delay_time>
<framed_protocol>PPP</framed_protocol>
<started xsi:type="xs:boolean">true</started>
<stopped xsi:type="xs:boolean">false</stopped>
</sessionParameters>
```

Removing Stale Sessions

Some devices, such as wireless LAN controllers (WLCs), may allow stale sessions to linger. In such cases, you use the HTTP DELETE API call to manually delete the inactive sessions. To do so, use cURL, a free 3rd-party command line tool for transferring data with URL (HTTP, HTTPS) syntax, as shown in the following procedure.

Cisco ISE no longer tracks stale sessions. This is to mitigate cases when Cisco ISE loses connectivity to the network for an extended period of time and misses accounting stops from the WLC or network access device (NAD). You can clear such stale information from Cisco ISE using the HTTP DELETE API call.

Note GNU Wget, the free utility for retrieving files using HTTP and HTTPS, does not support the HTTP DELETE API call.

Step 1 Log in to the target Monitoring node from the command line.

Note API calls are case sensitive and must be entered carefully. The variable <mntnode> represents the target Monitoring node.

Step 2 To manually delete a stale session for a MAC address, issue the following API call on the command line:

```
curl -X DELETE https://<mntnode>/ise/mnt/Session/Delete/MACAddress/<macaddress>
```

Step 3 To manually delete a stale session for a session ID, issue the following API call on the command line:

```
curl -X DELETE https://<mntnode>/ise/mnt/Session/Delete/SessionID/<sid#>
```

Step 4 To manually delete all sessions on the Monitoring node, issue the following API call on the command line:

```
curl -X DELETE https://<mntnode>/ise/mnt/Session/Delete/All
```

Using API Calls for Troubleshooting

Cisco ISE troubleshooting API calls send status requests to a targeted Monitoring node to retrieve the following diagnostic-related information:

- Node version and type (using a Version API call)
- Failure reasons (using a FailureReasons API call)
- Authentication status (using an AuthStatus API call)
- Accounting status (using an AcctStatus API call)

The following sections describe each type of troubleshooting API call as well as provide file examples, procedures for issuing each call, and a sample of the data returned:

Using Version API Calls

You use a Version API call to test the Representational State Transfer (REST) programming interface (PI) service and the credentials of each node. This section provides a schema file output example, a procedure for requesting the version of the Cisco ISE software and the node type by invoking this API call, and a sample of the node version and type that is returned after this API call is issued.

Each node type has an associated value and can be one of the following:

- STANDALONE_MNT_NODE = 0
- ACTIVE_MNT_NODE = 1
- BACKUP_MNT_NODE = 2
- NOT_AN_MNT_NODE = 3

Version API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="product" type="product"/>

  <xs:complexType name="product">
```

```
    <xs:sequence>
      <xs:element name="version" type="xs:string" minOccurs="0"/>
  <xs:element name="type_of_node" type="xs:int"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"/>
  </xs:complexType>
</xs:schema>
```

Invoking a Version API Call

Note Make sure that the target node is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the Version API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/<specific-api-call>):

```
https://acme123/ise/mnt/Version
```

Note You must carefully enter each API call in the URL address field of a target node because these calls are case sensitive. The use of "mnt" in the API call convention represents a Monitoring node.

Step 3 Press **Enter** to issue the API call.

Version API Call Data

In the following example, the Version API call returned this information about the targeted node:

- Node version—the example displays 1.0.3.032.
- Type of Monitoring node—the example displays 1, which means the node is active.

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<product name="Cisco Identity Services Engine">
<version>1.0.3.032</version>
<type_of_node>1</type_of_node>
</product>
```

Using FailureReasons API Calls

You use a FailureReasons API call to return a list of failed operations and possible resolutions, which are outlined in Table 3-1.

Note For details about using the Cisco ISE Failure Reasons Editor to access a complete list of operation failures, see Using the Failure Reasons Report, page A-1.

Table 3-1 Data Returned from FailureReasons API Calls

Element	Example
Failure reason ID	`<failureReason id="11011">`
Code	`<11011 RADIUS listener failed>`
Cause	`<Could not open one or more of the ports used to receive RADIUS requests>`
Resolution	`<Ensure that ports 1812, 1813, 1645 and 1646 are not being used by another process on the system>`

FailureReasons API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="failureReasonList" type="failureReasonList"/>

  <xs:complexType name="failureReasonList">
    <xs:sequence>
      <xs:element name="failureReason" type="failureReason" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="failureReason">
    <xs:sequence>
      <xs:element name="code" type="xs:string" minOccurs="0"/>
      <xs:element name="cause" type="xs:string" minOccurs="0"/>
      <xs:element name="resolution" type="xs:string" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"/>
  </xs:complexType>
</xs:schema>
```

Invoking a FailureReasons API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the FailureReasons API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/<specific-api-call>):

```
https://acme123/ise/mnt/FailureReasons
```

Note You must carefully enter each API call in the URL address field of a target node because the calls are case sensitive. The use of "mnt" in the API call convention represents a Monitoring node.

Step 3 Press **Enter** to issue the API call.

FailureReasons API Call Data

Note The following FailureReasons API call example displays a small sample of data that can be returned.

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<failureReasonList>
-
<failureReason id="100001">
-
<code>
100001 AUTHMGR-5-FAIL Authorization failed for client
</code>
<cause>This may or may not be indicating a violation</cause>
-
<resolution>
Please review and resolve according to your organization's policy
</resolution>
</failureReason>
-
<failureReason id="100002">
-
<code>
100002 AUTHMGR-5-SECURITY_VIOLATION Security violation on the interface
</code>
<cause>This may or may not be indicating a violation</cause>
-
<resolution>
Please review and resolve according to your organization's policy
</resolution>
</failureReason>
-
<failureReason id="100003">
-
<code>
100003 AUTHMGR-5-UNAUTHORIZED Interface unauthorized
</code>
```

```
<cause>This may or may not be indicating a violation</cause>
-
<resolution>
Please review and resolve according to your organization's policy
</resolution>
</failureReason>
-
<failureReason id="100004">
-
<code>
100004 DOT1X-5-FAIL Authentication failed for client
</code>
<cause>This may or may not be indicating a violation</cause>
-
<resolution>
Please review and resolve according to your organization's policy
</resolution>
</failureReason>
-
<failureReason id="100005">
<code>100005 MAB-5-FAIL Authentication failed for client</code>
<cause>This may or may not be indicating a violation</cause>
-
<resolution>
Please review and resolve according to your organization's policy
</resolution>
</failureReason>
-
<failureReason id="100006">
-
<code>
100006 RADIUS-4-RADIUS_DEAD RADIUS server is not responding
</code>
<cause>This may or may not be indicating a violation</cause>
-
<resolution>
Please review and resolve according to your organization's policy
</resolution>
</failureReason>
-
<failureReason id="100007">
-
<code>
100007 EPM-6-POLICY_APP_FAILURE Interface ACL not configured
</code>
<cause>This may or may not be indicating a violation</cause>
-
<resolution>
Please review and resolve according to your organization's policy
</resolution>
</failureReason>
```

For more information about the Cisco ISE Failure Reasons Editor, see Appendix A, “Using the Failure Reasons Report”.

Using AuthStatus API Calls

You use an AuthStatus API call to check the authentication status of sessions on a target node. You must specify at least one MAC address for queries.

This section contains the following sections:

You can configure the following search-related parameters:

- Duration—Defines the number of seconds in which an attempt is made to search and retrieve the authentication status records associated with a designated MAC address. Valid user-configurable values range from 1 to 864000 seconds (10 days). If you enter a value of 0 seconds, a default duration of 10 days is specified.
- Records—Defines the number of session records to be searched per MAC address. Valid user-configurable values range from 1 to 500 records. If you enter 0, a default setting of 200 records is specified.

  Note If you specify the value 0 for both the duration and the records parameters, the AuthStatus API call returns only the latest authentication session record associated with the designated MAC address.

  Here is an example of the generic form of a URL with the Duration and Records attributes:

  https://10.10.10.10/ise/mnt/AuthStatus/MACAddress/01:23:45:67:89:98/900000/2/All

- Attributes—Defines the number of attributes in the authentication status table that are returned from an authentication status search using the AuthStatus API call. Valid values include 0 (the default), All, or user_name+acs_timestamp (see the AuthStatus schema example, AuthStatus API Call Schema File, page 3-7).
  - If you enter “0”, the attributes defined in Table 3-2 are returned. These are listed in the restAuthStatus section of the output schema.
  - If you enter “All”, a fuller set of attributes are returned. These are listed in the fullRESTAuthStatus section of the output schema.
  - If you enter the values listed in the schema for user_name+acs_timestamp, only those attributes are returned. The user_name and acs_timestamp attributes are listed in the restAuthStatus section of the output schema.

Table 3-2 Authentication Status Table Attributes

Attribute	Description
name = “passed” or name = “failed”	Authentication status results: • Passed • Failed
name = “user_name”	Username
name = “nas_ip_address”	IP address/hostname for the network access device
name = “failure_reason”	Reason for session authentication failure

Table 3-2 Authentication Status Table Attributes (continued)

Attribute	Description
name = “calling_station_id”	Source IP address
name = “nas_port”	Network access server port
name = “identity_group”	Logical group consisting of related users and hosts
name = “network_device_name”	Name of the network device
name = “acs_server”	Name of the Cisco ISE appliance
name = “eap_authentication”	Extensible Authentication Protocol (EAP) method used for authentication request
name = “framed_ip_address”	Address configured for a specific user
name = “network_device_groups”	Logical group consisting of related network devices
name = “access_service”	Applied access service
name = “acs_timestamp”	Time stamp associated with the Cisco ISE authentication request
name = “authentication_method”	Identifies the method used in authentication
name = “execution_steps”	List of message codes for each diagnostic message logged while processing the request
name = “radius_response”	Type of RADIUS response (for example, VLAN or ACL)
name = “audit_session_id”	ID of the authentication session
name = “nas_identifier”	Network access server (NAS) associated with a specific resource
name = “nas_port_id”	ID of the NAS port used
name = “nac_policy_compliance”	Reflects Posture status (compliant or non compliant)
name = “selected_azn_profiles”	Identifies the profile used in authorization
name = “service_type”	Indicates a framed user
name = “eap_tunnel”	Tunnel or outer method used for EAP authentication
name = “message_code”	Identifies the audit message that defines the processed request result
name = “destination_ip_address”	Identifies the destination IP address

AuthStatus API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="authStatusOutputList" type="fullRESTAuthStatusOutputList"/>

  <xs:complexType name="fullRESTAuthStatusOutputList">
    <xs:sequence>
      <xs:element name="authStatusList" type="fullRESTAuthStatusList" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="fullRESTAuthStatusList">
    <xs:sequence>
      <xs:element name="authStatusElements" type="fullRESTAuthStatus" minOccurs="0"
maxOccurs="unbounded"/>
```

```
    </xs:sequence>
    <xs:attribute name="key" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="fullRESTAuthStatus">
    <xs:complexContent>
      <xs:extension base="restAuthStatus">
        <xs:sequence>
          <xs:element name="id" type="xs:long" minOccurs="0"/>
          <xs:element name="acsview_timestamp" type="xs:dateTime" minOccurs="0"/>
          <xs:element name="acs_session_id" type="xs:string" minOccurs="0"/>
          <xs:element name="service_selection_policy" type="xs:string" minOccurs="0"/>
          <xs:element name="authorization_policy" type="xs:string" minOccurs="0"/>
          <xs:element name="identity_store" type="xs:string" minOccurs="0"/>
          <xs:element name="response" type="xs:string" minOccurs="0"/>
          <xs:element name="cts_security_group" type="xs:string" minOccurs="0"/>
          <xs:element name="use_case" type="xs:string" minOccurs="0"/>
          <xs:element name="cisco_av_pair" type="xs:string" minOccurs="0"/>
          <xs:element name="ad_domain" type="xs:string" minOccurs="0"/>
          <xs:element name="acs_username" type="xs:string" minOccurs="0"/>
          <xs:element name="radius_username" type="xs:string" minOccurs="0"/>
          <xs:element name="nac_role" type="xs:string" minOccurs="0"/>
          <xs:element name="nac_username" type="xs:string" minOccurs="0"/>
          <xs:element name="nac_posture_token" type="xs:string" minOccurs="0"/>
          <xs:element name="nac_radius_is_user_auth" type="xs:string" minOccurs="0"/>
          <xs:element name="selected_posture_server" type="xs:string" minOccurs="0"/>
          <xs:element name="selected_identity_store" type="xs:string" minOccurs="0"/>
          <xs:element name="authentication_identity_store" type="xs:string"

minOccurs="0"/>
          <xs:element name="azn_exp_pol_matched_rule" type="xs:string" minOccurs="0"/>
          <xs:element name="ext_pol_server_matched_rule" type="xs:string" minOccurs="0"/>
          <xs:element name="grp_mapping_pol_matched_rule" type="xs:string" minOccurs="0"/>
          <xs:element name="identity_policy_matched_rule" type="xs:string" minOccurs="0"/>
          <xs:element name="nas_port_type" type="xs:string" minOccurs="0"/>
          <xs:element name="query_identity_stores" type="xs:string" minOccurs="0"/>
          <xs:element name="sel_exp_azn_profiles" type="xs:string" minOccurs="0"/>
          <xs:element name="selected_query_identity_stores" type="xs:string"
minOccurs="0"/>
          <xs:element name="tunnel_details" type="xs:string" minOccurs="0"/>
          <xs:element name="cisco_h323_attributes" type="xs:string" minOccurs="0"/>
          <xs:element name="cisco_ssg_attributes" type="xs:string" minOccurs="0"/>
          <xs:element name="other_attributes" type="xs:string" minOccurs="0"/>
          <xs:element name="response_time" type="xs:long" minOccurs="0"/>
          <xs:element name="nad_failure" type="xs:anyType" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="restAuthStatus">
    <xs:sequence>
      <xs:element name="passed" type="xs:anyType" minOccurs="0"/>
      <xs:element name="failed" type="xs:anyType" minOccurs="0"/>
      <xs:element name="user_name" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_ip_address" type="xs:string" minOccurs="0"/>
      <xs:element name="failure_reason" type="xs:string" minOccurs="0"/>
      <xs:element name="calling_station_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_port" type="xs:string" minOccurs="0"/>
      <xs:element name="identity_group" type="xs:string" minOccurs="0"/>
      <xs:element name="network_device_name" type="xs:string" minOccurs="0"/>
      <xs:element name="acs_server" type="xs:string" minOccurs="0"/>
      <xs:element name="eap_authentication" type="xs:string" minOccurs="0"/>
      <xs:element name="framed_ip_address" type="xs:string" minOccurs="0"/>
```

```
      <xs:element name="network_device_groups" type="xs:string" minOccurs="0"/>
      <xs:element name="access_service" type="xs:string" minOccurs="0"/>
      <xs:element name="acs_timestamp" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="authentication_method" type="xs:string" minOccurs="0"/>
      <xs:element name="execution_steps" type="xs:string" minOccurs="0"/>
      <xs:element name="radius_response" type="xs:string" minOccurs="0"/>
      <xs:element name="audit_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_identifier" type="xs:string" minOccurs="0"/>
      <xs:element name="nas_port_id" type="xs:string" minOccurs="0"/>
      <xs:element name="nac_policy_compliance" type="xs:string" minOccurs="0"/>
      <xs:element name="selected_azn_profiles" type="xs:string" minOccurs="0"/>
      <xs:element name="service_type" type="xs:string" minOccurs="0"/>
      <xs:element name="eap_tunnel" type="xs:string" minOccurs="0"/>
      <xs:element name="message_code" type="xs:string" minOccurs="0"/>
      <xs:element name="destination_ip_address" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Invoking a AuthStatus API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the AuthStatus API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/<specific-api-call>/MACAddress/<macaddress>/<seconds>/<numberofrecordspermacaddress>/All):

```
https://acme123/ise/mnt/AuthStatus/MACAddress/00:50:56:10:13:02/120/100/All
```

Note API calls are case sensitive. The use of "mnt" in the API call convention represents the target Monitoring node.

Step 3 Press **Enter** to issue the API call.

AuthStatus API Call Data

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<authStatusOutputList>
-
<authStatusList key="00:0C:29:46:F3:B8"><authStatusElements>
-
<passed xsi:type="xs:boolean">true</passed>
<failed xsi:type="xs:boolean">false</failed>
```

```
<user_name>suser77</user_name>
<nas_ip_address>10.77.152.209</nas_ip_address>
<calling_station_id>00:0C:29:46:F3:B8</calling_station_id>
<identity_group>User Identity Groups:Guest</identity_group>
<acs_server>guest-240</acs_server>
<acs_timestamp>2012-10-05T10:50:56.515Z</acs_timestamp>
<execution_steps>5231</execution_steps>
<message_code>5231</message_code>
<id>1349422277270561</id>
<acsview_timestamp>2012-10-05T10:50:56.517Z</acsview_timestamp>
<identity_store>Internal Users</identity_store>
<response_time>146</response_time>
<other_attributes>ConfigVersionId=81,EndPointMACAddress=00-0C-29-46-F3-B8,PortalName=Defau
ltGuestPortal,
CPMSessionID=0A4D98D1000001F26F0C04D9,CiscoAVPair=</other_attributes>
</authStatusElements>
-
<authStatusElements>
<passed xsi:type="xs:boolean">true</passed>
<failed xsi:type="xs:boolean">false</failed>
<user_name>00:0C:29:46:F3:B8</user_name>
<nas_ip_address>10.77.152.209</nas_ip_address>
<calling_station_id>00:0C:29:46:F3:B8</calling_station_id>
<identity_group>Guest_IDG</identity_group>
<network_device_name>switch</network_device_name>
<acs_server>guest-240</acs_server>
<authentication_method>mab</authentication_method>
<authentication_protocol>Lookup</authentication_protocol>
<acs_timestamp>2012-10-05T10:49:47.915Z</acs_timestamp>
<execution_steps>11001,11017,11027,15049,15008,15048,15048,15004,15041,15006,15013,24209,2
421
1,22037,15036,15048,15004,15016,11022,11002</execution_steps>
<response>{UserName
=00:0C:29:46:F3:B8; User-Name=00-0C-29-46-F3-B8;
State=ReauthSession:0A4D98D1000001F26F0C04D9;
Class=CACS:0A4D98D1000001F26F0C04D9:guest-240/138796808/76;
Termination-Action=RADIUS-Request; Tunnel-Type=(tag=1) VLAN;
Tunnel-Medium-Type=(tag=1) 802; Tunnel-Private-Group-ID=(tag=1) 2;
cisco-av-pair=url-redirect-acl=ACL-WEBAUTH-REDIRECT;
cisco-av-pair=url-redirect=https://guest-240.cisco.com:8443/guestportal/gateway?
sessionId=0A4D98D1000001F26F0C04D9&action=cwa;
cisco-av-pair=ACS:CiscoSecure-Defined-ACL=#ACSACL#-IP-pre-posture-506e980a;
cisco-av-pair=profile-name=WindowsXP-Workstation;}</response
><audit_session_id>0A4D98D1000001F26F0C04D9</audit_session_id><nas_po
rt_id>GigabitEthernet1/0/17</nas_port_id><posture_status>Pending</posture_status>
<selected_azn_profiles>CWA_Redirect</selected_azn_profiles>
<service_type>Call Check</service_type>
<message_code>5200</message_code>
<nac_policy_compliance>Pending</nac_policy_compliance>
<id>1349422277270556</id>
<acsview_timestamp>2012-10-05T10:49:47.915Z</acsview_timestamp>
<identity_store>Internal Endpoints</identity_store>
<response_time>13</response_time>
<other_attributes>ConfigVersionId=81,DestinationPort=1812,Protocol=Radius,AuthorizationPol
icyMatchedRule=CWA_Redirect,
NAS-Port=50117,Framed-MTU=1500,NAS-Port-Type=Ethernet,EAP-Key-N
ame=,cisco-nas-port=GigabitEthernet1/0/17,AcsSessionID=guest-240/138796808/76,Us
eCase=Host Lookup,SelectedAuthenticationIdentityStores=Internal
Endpoints,ServiceSelectionMatchedRule=MAB,IdentityPolicyMatchedRule=Default,CPMS
essionID=0A4D98D1000001F26F0C04D9,EndPointMACAddress=00-0C-29-46-F3-B8,EndPointM
atchedProfile=WindowsXP-Workstation,ISEPolicySetName=Default,HostIdentityGroup=E
ndpoint Identity Groups:Guest_IDG,Device Type=Device Type#All Device
Types,Location=Location#All Locations,Device IP
Address=10.77.152.209,Called-Station-ID=00:24:F7:73:9A:91,CiscoAVPair=audit-sess
```

```
ion-id=0A4D98D1000001F26F0C04D9</other_attributes>
-
</authStatusElements>
-
</authStatusList>
-
</authStatusOutputList>
```

Using AcctStatus API Call Data

You use an AcctStatus API call to retrieve the latest device and session account information on a target node. This section contains the following sections:

- AcctStatus API Call Schema File, page 3-11
- Invoking an AcctStatus API Call, page 3-12
- AcctStatus API Call Data, page 3-13

You can configure the following time-related parameter:

- Duration—Defines the number of seconds in which an attempt is made to search and retrieve the latest account device records associated with a designated MAC address. Valid user-configurable values range from 1 to 432000 seconds (5 days).
  - If you enter a value of 2400 seconds (40 minutes), this means that you want the device records for the designated MAC address that are available in the past 40 minutes.
  - If you enter a value of 0 seconds, this specifies a default duration of 15 minutes (900 seconds). This means that you want the device records for the designated MAC address that are available within the last 15 minutes.

An AcctStatus API call provides the following account status data fields as API outputs:

Table 3-3 Account Status Data Fields

Data Field	Description
MAC address	MAC address of the client
audit-session-id	Authentication session ID
Packets in	Total Packets received
Packets out	Total Packets transmitted
Bytes in	Total Bytes received
Bytes out	Total Bytes transmitted
Session time	Duration of current sessions

AcctStatus API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="acctStatusOutputList" type="restAcctStatusOutputList"/>

  <xs:complexType name="restAcctStatusOutputList">
    <xs:sequence>
      <xs:element name="acctStatusList" type="restAcctStatusList" minOccurs="0"
maxOccurs="unbounded"/>
```

```
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="restAcctStatusList">
    <xs:sequence>
      <xs:element name="acctStatusElements" type="restAcctStatus" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="macAddress" type="xs:string"/>
    <xs:attribute name="username" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="restAcctStatus">
    <xs:sequence>
      <xs:element name="calling_station_id" type="xs:string" minOccurs="0"/>
      <xs:element name="audit_session_id" type="xs:string" minOccurs="0"/>
      <xs:element name="paks_in" type="xs:long" minOccurs="0"/>
      <xs:element name="paks_out" type="xs:long" minOccurs="0"/>
      <xs:element name="bytes_in" type="xs:long" minOccurs="0"/>
      <xs:element name="bytes_out" type="xs:long" minOccurs="0"/>
      <xs:element name="session_time" type="xs:long" minOccurs="0"/>
      <xs:element name="username" type="xs:string" minOccurs="0"/>
  <xs:element name="server" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Invoking an AcctStatus API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a Cisco ISE node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the AcctStatus API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/<specific-api-call>/MACAddress/<macaddress>/<durationofcurrenttime>):

```
https://acme123/ise/mnt/AcctStatus/MACAddress/00:26:82:7B:D2:51/1200
```

Note You must carefully enter each API call in the URL address field of a target node because these calls are case sensitive. The use of "mnt" in the API call convention represents a Monitoring node.

Step 3 Press **Enter** to issue the API call.

AcctStatus API Call Data

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<acctStatusOutputList>
-
<acctStatusList macAddress="00:25:9C:A3:7D:48">
-
<acctStatusElements>
<calling_station_id>00:25:9C:A3:7D:48</calling_station_id>
<audit_session_id>0acb6b0b0000000B4D0C0DBD</audit_session_id>
<paks_in>0</paks_in>
<paks_out>0</paks_out>
<bytes_in>0</bytes_in>
<bytes_out>0</bytes_out>
<session_time>240243</session_time>
<server>HAREESH-R6-1-PDP1</server>
</acctStatusElements>
</acctStatusList>
</acctStatusOutputList>
```

Using Change of Authorization REST APIs

Change of Authorization (CoA) calls send commands to a specified session on a targeted Monitoring node to perform the following:

- Session reauthentication (using a Reauth API call)
- Session disconnection (using a Disconnect API call).

The following sections describe each type of CoA API call as well as provide schema file examples, procedures for issuing each call, and a sample of the data returned:

Using Reauth API Calls

A Reauth API call sends a reauthentication command to a specified session. Each session has an associated value and can be one of the following:

- 0—REAUTH_TYPE_DEFAULT
- 1—REAUTH_TYPE_LAST
- 2—REAUTH_TYPE_RERUN

Reauth API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="remoteCoA" type="coAResult"/>
<xs:complexType name="coAResult">
    <xs:sequence>
      <xs:element name="results" type="xs:boolean" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="requestType" type="xs:string"/>
  </xs:complexType>
</xs:schema>
```

Invoking a Reauth API Call

Note Make sure that you have verified that the target node is a valid Monitoring node. To verify the persona of a node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the Reauth API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/CoA/<specific-api-call>/<macaddress>/<reauthtype>/<nasipaddress>/<destinationipaddress>):

```
https://acme123/ise/mnt/CoA/Reauth/server12/00:26:82:7B:D2:51/2/10.10.10.10
```

Note You must carefully enter each API call in the URL Address field of a target node because the calls are case sensitive. The use of "mnt" in the API call convention represents a Monitoring node.

Step 3 Press **Enter** to issue the API call.

Reauth API Call Data

A Reauth API call returns one of the following results:

- True, which indicates that the command was successfully executed.
- False, which means that the command was not executed.

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<remoteCoA requestType="reauth">
<results>true</results>
</remoteCoA>
```

Using Disconnect API Calls

A Disconnect API call sends a disconnect command to a specified session and port. Each port has an associated value and can be one of the following:

- 0—DYNAMIC_AUTHZ_PORT_DEFAULT
- 1—DYNAMIC_AUTHZ_PORT_BOUNCE
- 2—DYNAMIC_AUTHZ_PORT_SHUTDOWN

Disconnect API Call Schema File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="remoteCoA" type="coAResult"/>

  <xs:complexType name="coAResult">
    <xs:sequence>
      <xs:element name="results" type="xs:boolean" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="requestType" type="xs:string"/>
  </xs:complexType>
</xs:schema>
```

Invoking a Disconnect API Call

Note Make sure that the target node to which you are issuing an API call is a valid Monitoring node. To verify the persona of a node, see Verifying a Monitoring Node, page 1-2.

Step 1 Log in to the target Monitoring node.

For example, when you initially log in to a Monitoring node with the hostname acme123, the following URL address field is displayed:

```
https://acme123/admin/LoginAction.do#pageId=com_cisco_xmp_web_page_tmpdash
```

Step 2 Enter the Disconnect API call in the URL address field of the target node by replacing the "/admin/" component with the API call component (/ise/mnt/CoA/<Disconnect>/<serverhostname>/<macaddress>/<portoptiontype>/<nasipaddress>/<destinationipaddress>):

```
https://acme123/ise/mnt/CoA/Disconnect/server12/
00:26:82:7B:D2:51/2/10.10.10.10
```

Note You must carefully enter each API call in the URL address field of a target node because the calls are case sensitive. The use of "mnt" in the API call convention represents a Monitoring node.

Step 3 Press **Enter** to issue the API call.

Disconnect API Call Data

A Disconnect API call returns one of the following results:

- True, which indicates that the command was successfully executed.
- False, which means that the command was not executed.

```
This XML file does not appear to have any style information associated with it. The
document tree is shown below.

-
<remoteCoA requestType="reauth">
<results>true</results>
</remoteCoA>
```

PART 2

External RESTful Services API

CHAPTER 5

Introduction to External RESTful Services API

External RESTful Services is based on HTTPS and REST methodology and uses port 9060. This chapter provides guidelines and examples for using the External RESTful Services application programming interface (API) supported by Cisco ISE as well as related API calls used to perform Create, Read, Update, Delete (CRUD) operations.

These APIs provide an interface to the ISE configuration data by enabling users, endpoints, endpoint groups, identity groups and SGTs to perform CRUD operations on the ISE data.

The HTTPS port 9060 is closed by default. The first requirement to use the API is to enable External RESTful Services from the ISE CLI.

Note If you try to invoke External RESTful Services API calls prior to enabling from the CLI, you will receive a response status as 403- “forbidden”.

External RESTful Services has a debug logging category, which you can enable in Cisco ISE on the debug logging page. For more information, see the Debug Log Configuration Options section of *Cisco Identity Services Engine User Guide, Release 1.2*.

All Representational State Transfer (REST) operations are audited and logged in the system.

Related Topics

Enabling the External RESTful Services API, page 5-2

Data Validation

CRUD data sent to the server is validated with the same validation rules that Cisco ISE has for the GUI. All validation is centralized in a validation layer. All XML data being posted is validated against the schema.

Two types of validation occur: data validation and structural validation. Data validation validates the data to be Cisco ISE compliant, for example, mandatory fields, field length, types, and so on. While structural validation validates the schema. For example, fields order, names, and so on.

External RESTful Services Namespaces

You should maintain strict namespaces within resources names and Uniform Resource Identifiers (URIs), including:

- Internal user identities, endpoints, endpoint groups, and identity groups.
- Security Group Access (SGA) for Security Group Tag (SGT).
- External RESTful Services for all other External RESTful Services object resources, such as search results that appears in the response message.

The Accept/Content-Type headers should contain the following namespace:

```
application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major
version>.<minor version>+xml
```

For example: `application/vnd.com.cisco.ise.identity.internaluser.1.0+xml`

The request XML should contain the following namespace definition:

```
identity.ers.ise.cisco.com

sga.ers.ise.cisco.com
```

For example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ns:endpoint xmlns:ns="identity.ers.ise.cisco.com" id="id">

<group>Profiled</group>

…

</ns:endpoint>
```

Enabling the External RESTful Services API

Step 1 Run the command **application configure ise**.

Step 2 The following options are displayed on the screen:

```
[1]Reset Active Directory settings to defaults
[2]Display Active Directory settings
[3]Configure Active Directory settings
[4]Restart/Apply Active Directory settings
[5]Clear Active Directory Trusts Cache and restart/apply Active Directory settings
[6]Enable/Disable External RESTful Services API
[7]Reset M&T Session Database
[8]Rebuild M&T Unusable Indexes
[9]Purge M&T Operational Data
[10]Reset M&T Database
[11]Refresh M&T Database Statistics
[12]Display Profiler Statistics
[13]Exit
```

Step 3 Enter **6**and press **Enter**.

The following message appears:

```
Current External RESTful Services State: disabled

By proceeding, External RESTful Services port 9060 will be opened and External RESTful
Services API will be enabled
```

```
Are you sure you want to proceed? y/n [n]:
```

Step 4 Enter **y** and press **Enter**.

The following message appears:

```
Enabling External RESTful Services port 9060...

External RESTful Services API enabled
```

Step 5 Verify if the External RESTful Services API is enabled by accessing the External RESTful Services SDK page at the following URL: https://<ipaddress>:9060/ers/sdk. You should always add the port number as 9060 to access the SDK.

External RESTful Services Admin

You must create an ISE Administrator with the External RESTful Services Admin group in order to use the APIs. For more information on creating admin users, refer to the following section in the Cisco Identity Services User Guide, Release 1.2:

http://www.cisco.com/en/US/docs/security/ise/1.2/user_guide/ise_man_admin.html#wp1579129

REST Client

To Use the External RESTful Services API, an HTTPS client is required. You can use a curl command to post requests to the server, write your own Python scripts or Java clients. You can also use any HTTP posting tool such as the POSTMAN plugin from Chrome browser. Now External RESTful Services is enabled and ready, you can start using it.

Related Topics

Invoking an External RESTful Services API Call, page 6-1

Chapter 6, “Using a REST API Client”

Resource Version and MediaType

In External RESTful Services, resource representations and request bodies are normally encoded in XML (as specified in RFC4267). Each type of resource has its own media-type, which matches the following pattern:

application/vnd.com.cisco.ise.xxx.yyy.version+xml;charset=UTF-8

Where "xxx" represents the namespace, "yyy" the resource, and version specifies the resource version used by the client. (RFC 3023). For example, the MediaType for Internal User resource with schema version 1.0 is represented as follows:

application/vnd.com.cisco.ise.identity.internaluser.1.0+xml;charset=UTF-8

The External RESTful Services API must provide representations of all resources available in XML. Whenever the requested media type is not supported by the Cisco ISE server, status 415 will be returned with a list of causes such as "Resource version is no longer supported".

External RESTful Services Requests

In requests made to External RESTful Services, several specific HTTP headers are used as described in Table 5-1.

Table 5-1 External RESTful Services Request Headers

Header	Supported Values	Description of Use	Required
Accept	Comma-delimited list of media types or media type patterns.	Indicates to the server what media types this client is prepared to accept including the resource version.	Yes, on GET/GET ALL/DELETE/GET VERSION operations (these contain no message body).
Authorization	“Basic” plus username and password (per RFC 2617).	Identifies the authorized user making this request.	Yes, on all requests.
Content-Length	Length (in bytes) of the request message body.	Describes the size of the message body.	Yes, on requests that contain a message body.
Content-Type	Media type describing the request message body.	Describes the representation and syntax of the request message body.	Yes, on requests that contain a message body.

External RESTful Services Response Headers

In responses returned by the External RESTful Services, several specific HTTP headers are used as described in Table 5-2.

Table 5-2 External RESTful Services Response Headers

Header	Supported Values	Description of Use	Required
Content-Length	Length (in bytes) of the response message body.	Describes the size of the message body.	Yes, on responses that contain a message body.
Content-Type	Media type describing the response message body.	Describes the representation and syntax of the response message body.	Yes, on responses that contain a message body.
Location	Canonical URI of a newly created resource.	Returns a new URI that can be used to request a representation of the newly created resource.	Yes, on responses to requests that create new server side resources accessible via a URI.

Common External RESTful Services HTTP Response Codes

External RESTful Services returns common HTTP response codes as described in Table 5-3. In addition to the status codes returned in the response header, each request might have additional XML content according to the nature of the request.

Table 5-3 Description of HTTP Response Codes Returned By External RESTful Services

HTTP Status	Description
200 OK	The request was successfully completed. If this request created a new resource that is addressable with a URI, and a response body is returned containing a representation of the new resource, a 200 status will be returned with a Location header containing the canonical URI for the newly created resource.
201 Created	A request that created a new resource was completed, and no response body containing a representation of the new resource is being returned. A Location header containing the canonical URI for the newly created resource should also be returned.
202 Accepted	The request has been accepted for processing, but the processing has not been completed. Per the HTTP/1.1 specification, the returned entity (if any) *should* include an indication of the current status of the request and either a pointer to a status monitor or some estimate of when the user can expect the request to be fulfilled.
204 No Content	The server fulfilled the request but does not need to return a response message body.
400 Bad Request	The request could not be processed because it contains missing or invalid information (such as a validation error on an input field or a missing required value).
401 Unauthorized	The authentication credentials included with the request are missing or invalid.
403 Forbidden	The server recognized your credentials, but you do not possess authorization to perform this request.
404 Not Found	The request specified a URI of a resource that does not exist.
405 Method Not Allowed	The HTTP verb specified in the request (DELETE, GET, HEAD, POST, PUT) is not supported for this request URI.
406 Not Acceptable	The resource identified by this request is not capable of generating a representation corresponding to one of the media types in the Accept header of the request.
409 Conflict	A creation or update request could not be completed, because it would cause a conflict in the current state of the resources supported by the server (for example, an attempt to create a new resource with a unique identifier already assigned to some existing resource).
415 Unsupported Media Type	The media type specified in the Accept header is not supported by the server. This will be the common response when the client resources version is no longer supported by the server.
429 Too many requests	There are too many simultaneous External RESTful Services requests.
500 Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.
501 Not Implemented	The server does not (currently) support the functionality required to fulfill the request.
503 Service Unavailable	The server is currently unable to handle the request due to temporary overloading or maintenance on the server.

Version Control with the External RESTful Services API

The External RESTful Services API provides backward compatibility with previous Cisco ISE versions through a versioning mechanism. Because Cisco ISE, Release 1.2, is the first External RESTful Services release, all resources are version 1.0 and no backward compatibility is required.

Each RESTful resource has a model version (major.minor). The version must be part of the request header with the syntax as follows:

```
application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major version>.<minor version>+xml
```

For example, to get internal user resource version 1.0, the following request is passed:

```
DELETE /ers/config/internaluser/333 HTTP/1.1
Host: cisco.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
```

After authenticating and authorizing the request, a version-match check is performed with one of the following matching results:

Table 5-4 Version-match Results

Version-match	Outcome
No version sent	The server returns status 415 “Unsupported Media Type”.
Client version equal to server version	The server proceeds with processing the request.
Client minor version not equal to server minor version	The server adds a response warning message describing the versions gap and proceeds processing the request.
Client and server major version does not match	The server returns status 415 with a corresponding error message.

In addition, each resource has an API to retrieve a list of server-supported versions.

Paging with the External RESTful Services API

The search results by default are paged to 20 resources per page. Page numbering starts at page number 0. The maximum number of resources per page cannot exceed 100. You can override the defaults by using the paging parameters. Paging parameter are passed in the URI using query parameters.

For example, to get the first 50 records of an internal user sorted in ascending order by the “name” field, the following request is passed:

```
GET/ers/config/internaluser?page=0&size=50&sortacs=name.EQ.Finance HTTP/1.1
```

The following paging parameters are available:

Table 5-5 Paging Parameters

Parameter	Description
page	Page start index. Default value is 0.
size	Page size. Default value is 10, maximum page size is 100.
sortbyasc	Sort field in ascending order. Default value is the “name” field.
sortbydsc	Sort field in descending order. Default value is the “name” field.

Sorting with External RESTful Services API

By default, the search results are sorted according to the column name in an ascending order. You can override the default sort settings by specifying the sorting parameters. You can pass the sorting parameters in the URI using query parameters. You can specify 'sortasc' (ascending order) or 'sortdsc' (descending order) parameters to override the default settings.

For example, to sort the first 50 records of an internal user in a descending order by the “name” field, the following request is passed:

```
GET
/ers/config/internaluser?filter=name.STARTW.a&filter=identityGroup.EQ.Finance&size=50&p
age=0&sortdsc=name
```

Filtering with External RESTful Services API

You can perform simple filtering operations through the filter query string parameter. You can send more than one filter. The logical operator common to all filter criteria is AND by default. You can change this by using the “filtertype=or” query string parameter.

Each resource data model description should specify if an attribute is a filtered field.

For example, to get internal users with a first name starting with ‘a’ and belonging to the ‘Finance’ identity group, the following request is passed:

```
GET
/ers/config/internaluser?page=0&size=20&sortacs=name&filter=name.STARTSW.a&filter=identity
Group.EQ.Finance HTTP/1.1
```

The following filter parameters are available:

Table 5-6 Available Filter Parameters

Parameter	Description
EQ	Equals
GT	Greater than
LT	Less than
STARTSW	Start with
ENDSW	End with
CONTAINS	Contains

Table 5-6 Available Filter Parameters (continued)

Parameter	Description
NEQ	Not equals
NSTARTSW	Not start with
NENDSW	Not end with
NCONTAINS	Not contains

Table 5-7 List of Filterable Attributes for Each Resource

Resource	Filterable Attributes
Endpoints	mac, portalUser, profile, profileId, staticGroupAssignment, staticProfileAssignment
Internal User	name
Endpoint Identity Group	name
Identity Groups	name
SGTs	name
Profiler Policy (Read-only)	name

Example of links within a search result

```
<ns2:searchResult xmlns:ns2="ers.ise.cisco.com"   total="1163">
<link rel="self"
href="http://cisco.com/ers/config/internaluser?page=0&size=20"
type="application/xml"/>
<link rel="next" href="http://cisco.com/ers/config/internaluser?page=20&size=20"
type="application/xml"/>
<resources>
<link rel="john doe" href="http://cisco.com/ers/config/internaluser/333"
type="application/xml"/>
<link rel="jeff smit" href="http://cisco.com/ers/config/internaluser/444"
type="application/xml"/>
.
.
.
</resources>
</ns2:searchResult>
```

External RESTful Services Data Model

External RESTful Services data model defines the representations of the RESTful resources that the External RESTful Services API operates up on. The representations are made up of fields, each with a name and value, encoded using an XML dictionary. The values are numeric or string literals, lists, or dictionaries, each of which are represented in XML.

Each type of resource contains its own internet media type. The internal media type must conform to the following pattern:

```
application/vnd.com.cisco.ise.resource.version+xml;charset=UTF-8
```

The media type corresponding to each RESTful resource is included in square brackets in the section header.

In the resource model descriptions, fields annotated with [POST] are included in a POST request that is used to create new resources. Similarly, fields annotated with [PUT] are included in a PUT request that is used to update properties of existing resources. You must not include fields that are not annotated in the request body of the PUT or POST requests. Such requests are ignored by the server.

Base Resource

Each resource contains a base representation that constitutes a set of attributes or fields mentioned in the following table:

Table 5-8 Base Representation Attributes

Field Name	Type	Default Value	Occurs	Description
URI	Hyperlink	—	1	A GET request made against this URI refreshes the client representation of the resources that are accessible to this user.
id	String	—	1	The resource uid [PUT]
name	String	—	1	The name of the resource[POST][PUT]
description	String	—	0..1	Description of the resource[POST][PUT]

Internal User

The internet media type for the internal user must conform to the following format:

```
application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
```

An internal user data model contains the set of attributes or fields mentioned in the following table:

Table 5-9 Internal User Data Model - Attributes

Field Name	Type	Default Value	Occurs	Description
URI	Hyperlink	—	1	A GET request made against this URI refreshes the client representation of the resources that are accessible to this user.
id	String	—	1	The resource uid [PUT]
name	String	—	1	The internal user name [POST][PUT]
description	String	—	0..1	Description of the user [POST][PUT]
enabled	boolean	true	1	Indicates if the user is enabled [POST][PUT]
email	String	—	0..1	The email address of the user [POST][PUT]
password	String	—	1	The password of the user [POST][PUT]
firstName	String	—	1	The first name of the user [POST][PUT]

Table 5-9 Internal User Data Model - Attributes (continued)

Field Name	Type	Default Value	Occurs	Description
lastName	String	—	1	The last name of the user [POST][PUT]
changePassword	boolean	true	1	Forces a password change up on the next login attempt
identityGroups	String	—	1	Comma separated string of identityGroup ids.
costumeAttributes	Map (K,V)	—	0..1	Map with Key String representing the name of the attribute and a value string representing the value of the attribute.

Endpoint

The internet media type for the internal user must conform to the following format:

`application/vnd.com.cisco.ise.identity.endpoint.1.0+xml`

An Endpoint data model contains the set of attributes or fields mentioned in the following table:

Table 5-10 Endpoint Data Model - Attributes

Field Name	Type	Default Value	Occurs	Description
URI	Hyperlink	—	1	A GET request made against this URI refreshes the client representation of the resources that are accessible to this endpoint.
id	String	—	1	The resource uid [PUT]
description	String	—	0..1	The description of the endpoint [POST][PUT]
mac	String	true	1	The Endpoint MAC Address
profileID	String	—	0..1	The profile ID
groupId	String	—	0..1	Comma separated string of identity group ids
StaticGroupAssignment	boolean	false	1	—
StaticProfileAssignment	boolean	false	1	—
portalUser	String	—	0..1	—
identityStore	String	—	—	—
identityStoreId	String	—	—	—

Endpoint Identity Group

The internet media type for the Endpoint identity group must conform to the following format:

`application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xm`

An Endpoint identity group data model contains the set of attributes or fields mentioned in the following table:

Table 5-11 Endpoint Identity Group Data Model - Attributes

Field Name	Type	Default Value	Occurs	Description
URI	Hyperlink	—	1	A GET request made against this URI refreshes the client representation of the resources accessible to this user.
id	String	—	1	The resource uid [PUT]
name	String	—	1	The name of the identity group [POST][PUT]
description	String	—	0..1	The description of the identity group [POST][PUT]
systemDefined	boolean	—	1	—

Identity Group

The internet media type for the identity group must conform to the following format:

`application/vnd.com.cisco.ise.identity.identitygroup.1.0+xml`

An identity group data model contains the set of attributes or fields mentioned in the following table:

Table 5-12 Identity Group Data Model - Attributes

Field Name	Type	Default Value	Occurs	Description
URI	Hyperlink	—	1	A GET request made against this URI refreshes the client representation of the resources that are accessible to this user.
id	String	—	1	The resource uid [PUT]
name	String	—	1	The name of the identity group [POST][PUT]
description	String	—	0..1	The description of the identity group [POST][PUT]

SGT

The internet media type for the SGTs must conform to the following format:

`application/vnd.com.cisco.ise.sga.sgt.1.0+xml`

An SGT data model contains the set of attributes or fields mentioned in the following table:

Table 5-13 SGT Data Model - Attributes

Field Name	Type	Default Value	Occurs	Description
URI	Hyperlink	—	1	A GET request made against this URI refreshes the client representation of the resources that are accessible to this user.
id	String	—	1	The resource uid [PUT]

Table 5-13 SGT Data Model - Attributes (continued)

Field Name	Type	Default Value	Occurs	Description
name	String	—	1	The name of the SGT [POST][PUT]
description	String	—	0..1	The description of the SGT [POST][PUT]
value	String	—	1	The SGT value
generatedId	String	—	1	The SGT generated Id. This attribute is read-only
isTagFromRange	boolean	—	1	isTagFromRange

VersionInfo

The internet media type for the VersionInfo data model must conform to the following format:

`application/vnd.com.cisco.ise.ers.versioninfo.1.0+xml`

A VersionInfo data model contains the set of attributes or fields mentioned in the following table:

Table 5-14 VersionInfo Data Model - Attributes

Field Name	Type	Occurs	Description
supportedVersions	String	1	The CSV that describes the version(s) supported by this data model.
currentServerVersion	String	1	The version of the server data model implementation.

SearchResult

The internet media type for the SearchResult data model must conform to the following format:

`application/vnd.com.cisco.ise.ers.searchresult.1.0+xml`

A SearchResult data model contains the set of attributes or fields mentioned in the following table:

Table 5-15 SearchResult Data Model - Attributes

Field Name	Type	Occurs	Description
total	String	1	Total number of results in the search
type	String	1	The resource type on which a search is performed
self	Hyperlink	1	A link to refresh current representation
next	Hyperlink	0..1	A link to get next results page
prev	Hyperlink	0..1	A link to previous results page
resources	BaseResource[]	0..1	A collection of base resources

External RESTful Services Response

The internet media type for the External RESTful Services Response data model must conform to the following format:

application/vnd.com.cisco.ise.ersresponse.1.0+xml

An External RESTful Service Response data model contains the set of attributes or fields mentioned in the following table:

Table 5-16 External RESTful Services Response Data Model - Attributes

Field Name	Type	Occurs	Description
operation	String	1	Describes the operation performed, for example, [put]update-internaluser
targetURI	Hyperlink	0..1	The request URI followed by that response
messages	ResponseMessage[]	0..1	Array of messages from the server

Response Message

The internet media type for the Response message data model must conform to the following format:

application/vnd.com.cisco.ise.ersresponse.1.0+xml

A Response message data model contains the set of attributes or fields mentioned in the following table:

Table 5-17 Response Message Data Model - Attributes

Field Name	Type	Occurs	Description
title	String	1	Localized text describing the nature of the problem reported by this message.
type	String	1	Describes the message type using the following values: • INFO • WARNING • ERROR
code	String	0..1	Symbolic error code identifying the type of error reported by this message, for example, validation error
stackTrace	String	0..1	Stack trace associated with this message (in debug mode only).
hint	String	0..1	Localized text further describing the nature of the problem, possibly including potential workarounds that the client could try.

Error Responses

When ever the request fails, a HTTP error status and response content is returned back to client in order to describe the problem. The following table describes possible errors:

Table 5-18 Error Codes

Error Code	Description	HTTP Status
ERS_INTERNAL_EXCEPTION	Any unexpected internal server error that occurs at runtime.	500
ERS_VERSION_EXCEPTION	Occurs when the resource version sent in the request content is not supported anymore by the server.	415
ERS_MEDIA_TYPE_EXCEPTION	Occurs when the media type sent by the client in the ACCEPT or Content-Type headers is invalid.	415
ERS_UNSUPPORTED_RESOURCE_EXCEPTION	Occurs when the resource as appear in the URI, does not supported by the server.	400
ERS_UNSUPPORTED_METHOD_EXCEPTION	Occurs when the request method type is not supported for the specified URI.	400
ERS_QUERY_VALIDATION_EXCEPTION	Occurs when the search filter or paging parameters are not valid.	400
ERS_SCHEMA_VALIDATION_EXCEPTION	Occurs when the resource validation against its schema fails.	400
ERS_CONVERSION_EXCEPTION	Occurs for some internal conversions and should be treated as INTERNAL_EXCEPTION.	500
ERS_APPLICATION_RESOURCE_VALIDATION_EXCEPTION	Occurs when the resource semantic validation does not meets the requirements.	400
ERS_CRUD_OPERATION_EXCEPTION	Occurs during the CRUD operation execution and should be treated as INTERNAL_EXCEPTION	500

Updated Fields

The internet media type for the Update fields data model must conform to the following format:

`application/vnd.com.cisco.ise.ers.updatedfields.1.0+xml`

A Update fields data model contains the set of attributes or fields mentioned in the following table:

Table 5-19 Update Fields Data Model - Attributes

Field Name	Type	Occurs	Description
field	String	1	The modified field name
oldValue	String	1	Field's old value
newValue	String	1	Field's modified value

Security Features in External RESTful Services API

The External RESTful Services API is disabled by default and a user with administrative privileges must explicitly enabled it. The External RESTful Services API supports only HTTPS access (using port 9060) and basic HTTP authentication. Plain HTTP access is not supported. The External RESTful Services API implementation employs a mechanism to thwart brute force attacks on the passwords.

External RESTful Services Authentication

The External RESTful Services API runs over HTTPS only and supports basic authentication. The authentication credentials are encrypted and are part of the request header. Authentication is implemented using the basic authentication mechanism corresponding to the Tomcat server.

Note As External RESTful Services applications are completely stateless, no session is managed.

External RESTful Services Authorization

External RESTful Services API provides read-only and full-access level authorization. An administrator must assign special privileges to a user to perform operations using the External RESTful Services. The following two roles can be assigned

- External RESTful Services Super User—For full access to External RESTful Services API (GET, POST, DELETE, PUT).
- External RESTful Services Guest—For read-only access (GET requests only).

If you do not have the required permissions and still try to perform External RESTful Services operations, you will receive an error response.

Related Topics

- Creating a New Cisco ISE Administrator

Injection Attacks

The External RESTful Services web application is secured against all kinds of injection attacks including Cross Side Scripting, SQL/HQL Injection, Shell Injection, and filename manipulation attacks. Sufficient input validation is also performed.

Brute Force Attacks

External RESTful Services API provides a mechanism to prevent brute force attacks on passwords. If any user breaks the password policy that is defined in the Cisco ISE GUI, such user profiles are suspended or disabled resulting in 401 status message.

Connection Limiting

The port that External RESTful Services use (https on port 9060) accepts only ten or less concurrent connections. This mechanism prevents clients from misusing the API (CPU starvation) and also from DOC Attacks.

External RESTful Services in an ISE Deployment

All External RESTful Services requests are valid only for the primary node. Secondary nodes have only read-access (GET requests).

External RESTful Services SDK

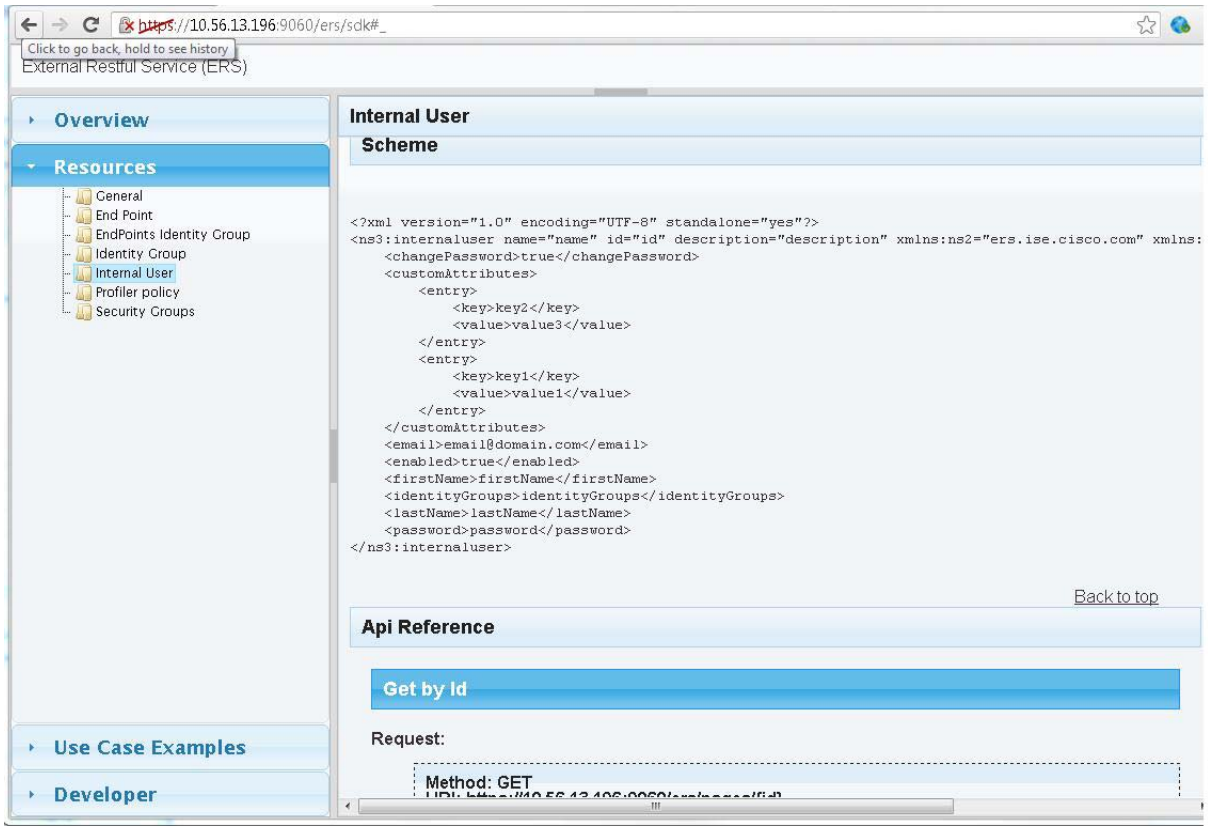
You can use the External RESTful Services software developer's kit (SDK) page to build your own tools. You access the page from the following URL: https://<ipaddress>:9060/ers/sdk

The External RESTful Services SDK page can be accessed by Admin users only. It contains the following information:

- List of all available API calls with xml examples including request/response structures
- Schema files available for download
- Cisco ISE API Reference Guide
- Java External RESTful Services Client demo application

See Figure 5-1 for all the functions available.

Figure 5-1 External RESTful Services SDK



Downloading External RESTful Services Schema Files

External RESTful Services SDK is shipped with the following XSD schema files that are supported on the External RESTful Services API:

- ers.xsd
- identity.xsd
- sga.xsd

Step 1 Log in to the External RESTful Services SDK page with the following URL:

https://<ipaddress>:9060/ers/sdk

Step 2 Log in as an External RESTful Services Admin.

Step 3 Click **Schema Files** available under **Downloads**.

You can use the files with available tools such as JAXB to generate schema classes.

You can develop HTTP client code or use any third-party HTTP client code and integrate it with the schema classes that are generated from the XSD files.

Note The XML sent in the content is validated against the schema. Therefore, field order and syntax in the XML should be the same as it appears in the schema. Otherwise, you will receive a bad request status code.

External RESTful Services System Flow

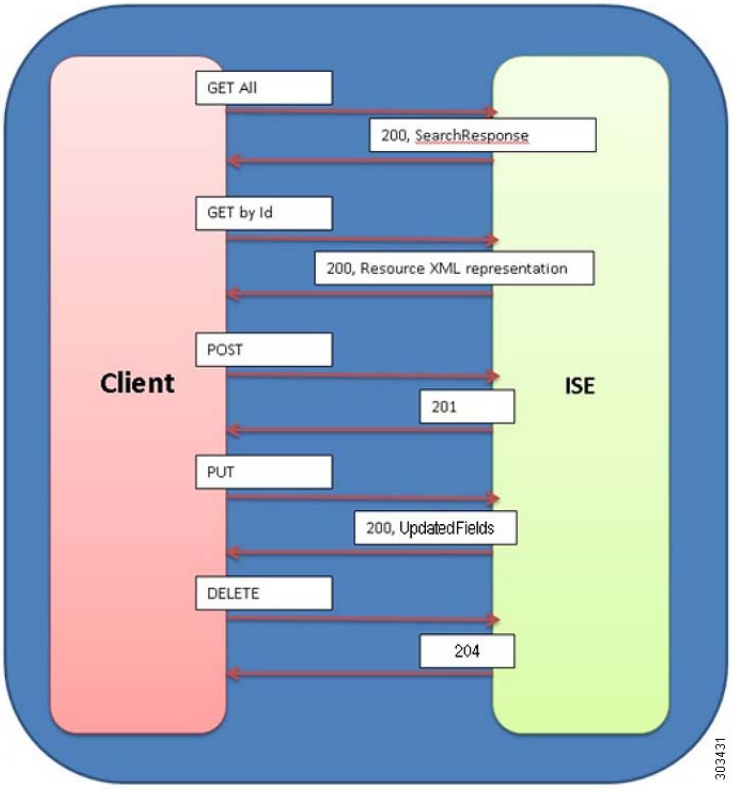
Common External RESTful Services flows always consist of an HTTPS request sent from a client and an HTTPS response from the server. The flows differ by request types, URIs, request headers, response headers, and response contents.

External RESTful Services Success Flow Sequence

Common flows will always consist of HTTPS requests sent from the client and an HTTPS response from the ISE server. The flows differ by request types, URI's, request headers, response headers and response contents. Successful requests will generally return an HTTP status code of 200 (OK), 201 (Created), or 204 (No Content), to indicate that the requested action has been successfully performed. In addition, they might include a response message body (with an appropriate media type) containing a representation of the requested information. However, it is possible for a number of things to go wrong. The various underlying causes are described by various HTTP status codes in the range 400-499 (for client side errors) or 500-599 (for server side problems). The description of each request type SHOULD list the possible status codes returned by that request type.

The following figure shows an example of an External RESTful Services success flow.

Figure 5-2 External RESTful Services Success Flow Sequence



Related Topics

Common External RESTful Services HTTP Response Codes, page 5-4

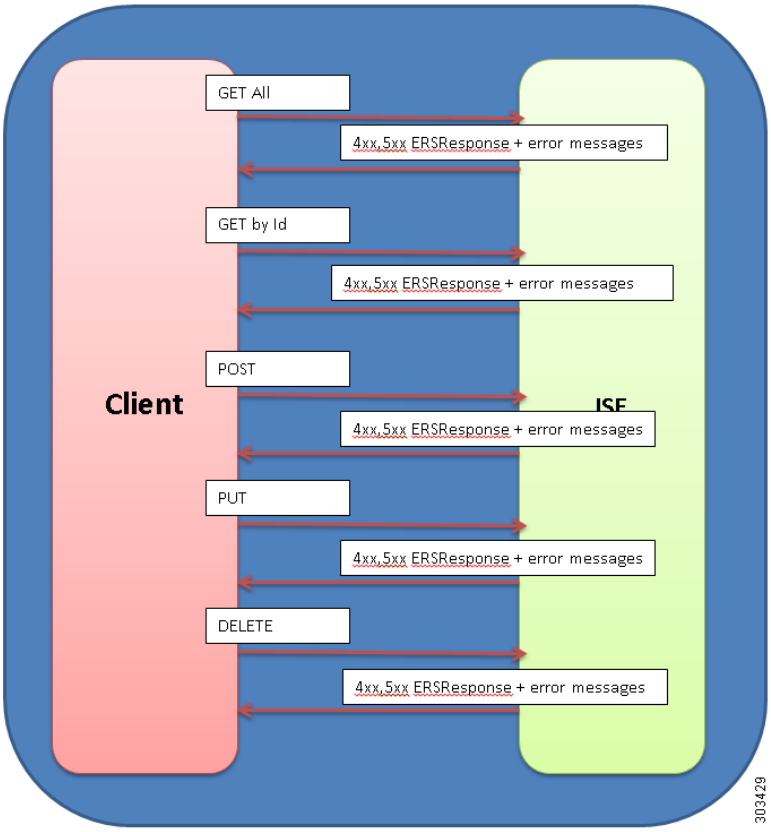
External RESTful Services Failure Flow Sequence

It is possible for a number of things to go wrong during the request processing. The various underlying causes are described by various HTTP status codes in the range 400-499 (for client side errors) or 500-599 (for server side problems). The description of each request type SHOULD list the possible status codes returned by that request type. If a response is returned with an error status code (400-499 or 500-599), the server SHOULD also return a response message body containing a messages data model, containing zero or more message data models, describing what went wrong. The text values of such messages might be used, for example, to communicate with a human user of the client side application.

In failed flow, the response content will contain a list of the error messages that caused the failure.

The following figure shows an example of an External RESTful Services failure flow.

Figure 5-3 External RESTful Services Failure Flow Sequence



Related Topics

Common External RESTful Services HTTP Response Codes, page 5-4

External RESTful Services Calls

Invoking an External RESTful Services API Call

You should fulfill the following prerequisites before invoking an External RESTful Services API call:

- Enable External RESTful Services from the CLI.
- Enable External RESTful Services Admin privileges.

You can use any REST client like JAVA, cURL Linux command, python to invoke External RESTful Services API calls.

Related Topics

- Enabling the External RESTful Services API, page 5-2
- External RESTful Services Authorization, page 5-15

GetVersion API Call

The GetVersion API call is common to all available resources: endpoints, internal users, endpoint identity groups, and security group tags (SGTs). It fetches the version information of the required resource.

Table 6-1 GetVersion Characteristics

Characteristics	Description
Description	Retrieve the version information of the specified resource
Synopsis	GET/ers/config/<resource-name>/versioninfo
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	Resource of type VersionInfo
Response Status	200, 400, 401,403,404,415,500

GetVersion Sample Request

```
Sample Request for Get Version Internal Users API
Method:GET
URI: https://10.56.13.196:9060/ers/config/internaluser/versioninfo
HTTP Accept header:
application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
```

GetVersion Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:versionInfo xmlns:ns2="ers.ise.cisco.com">
    <currentServerVersion>1.0</currentServerVersion>
    <link type="application/xml" href="link" rel="self"/>
    <supportedVersions>0.9,0.8</supportedVersions>
</ns2:versionInfo>
```

Internal User External RESTful Services API Calls

External RESTful Services API Calls for Internal users support full Create, Read, Update, Delete (CRUD) functionality. The internal user ID used in these API calls is the UUID type stored in the Cisco ISE database.

Table 6-2 External RESTful Services API Calls Available for Internal Users

API Call	HTTP Method	URL	Content	QueryString
Get All Users	GET	/ers/config/internaluser	—	Page, Size, sortacs or sortdsn, Filter
Get User	GET	/ers/config/internaluser/{internal user-id}	—	—
Create User	POST	/ers/config/internaluser/	internaluser	—
Update User	PUT	/ers/config/internaluser/{internal user-id}	internaluser	—
Delete User	DELETE	/ers/config/internaluser/{internal user-id}	—	—

Internal User Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:internaluser name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <changePassword>true</changePassword>
    <customAttributes>
        <entry>
            <key>key2</key>
            <value>value3</value>
        </entry>
        <entry>
            <key>key1</key>
            <value>value1</value>
        </entry>
    </customAttributes>
    <email>email@domain.com</email>
    <enabled>true</enabled>
    <firstName>firstName</firstName>
    <identityGroups>identityGroups</identityGroups>
    <lastName>lastName</lastName>
    <password>password</password>
</ns3:internaluser>
```

Get All Users API Call

You use Get All Users API call to retrieve all the internal users in Cisco ISE.

Table 6-3 Get All Users Characteristics

Characteristics	Description
Description	Retrieve collection of internal users
Synopsis	GET /ers/config/internaluser
Request Headers	Accept, Authorization, Host
QueryString	start, size, sortbyacn, sortbydcn, filter (fields supported by filter: name, description, firstName, lastName, identityGroup, email, enabled)
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	SearchResult
Response Status	200, 400, 401, 403, 404, 415, 500

Get All Users Sample Request

```
Method: GET
URI: https://10.56.13.196:9060/ers/config/internaluser
HTTP Accept header:
application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
Request Content:
N/A
```

Get All Users Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
    <nextPage type="application/xml" href="link-to-next-page" rel="next"/>
    <previousPage type="application/xml" href="link-to-previous-page" rel="previous"/>
    <resources>
        <resource name="name1" id="id1" description="description1"/>
        <resource name="name2" id="id2" description="description2"/>
    </resources>
</ns2:searchResult>
```

Get User API Call

You use the Get User API call to retrieve an internal user using the user ID.

Table 6-4 Get User Characteristics

Characteristics	Description
Description	Retrieve the specified internal user
Synopsis	GET /ers/config/internaluser/{internaluser-id}
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	Resource of type InternalUser
Response Status	200, 400, 401,403, 404, 415, 500

Get User Sample Request

```
Method: GET
URI: https://10.56.13.196:9060/ers/config/internaluser/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
Request Content:
N/A
```

Get User Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:internaluser name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <changePassword>true</changePassword>
    <customAttributes>
        <entry>
            <key>key2</key>
            <value>value3</value>
        </entry>
        <entry>
            <key>key1</key>
            <value>value1</value>
        </entry>
    </customAttributes>
    <email>email@domain.com</email>
    <enabled>true</enabled>
    <firstName>firstName</firstName>
    <identityGroups>identityGroups</identityGroups>
    <lastName>lastName</lastName>
    <password>password</password>
</ns3:internaluser>}
```

Create User API Call

You use the Create User API call to create internal users in Cisco ISE. A password is mandatory for creating an internal user using the External RESTful Services API.

Table 6-5 Create User API Call Characteristics

Characteristic	Description
Description	Create the specified internal user
Synopsis	POST /ers/config/internaluser/
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	InternalUser
Response Headers	Content-Length, Content-Type, Location
Response Message Body	Resource of type InternalUser
Response Status	201, 400, 401, 403, 415, 500

Create User Sample Request

```
Method: POST
URI: https://10.56.13.196:9060/ers/config/internaluser
HTTP Content-Type header:
application/vnd.com.cisco.ise.identity.internaluser.1.0+xml; charset=utf-8
Request Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:internaluser name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <changePassword>true</changePassword>
    <customAttributes>
        <entry>
            <key>key2</key>
            <value>value3</value>
        </entry>
        <entry>
            <key>key1</key>
            <value>value1</value>
        </entry>
    </customAttributes>
    <email>email@domain.com</email>
    <enabled>true</enabled>
    <firstName>firstName</firstName>
    <identityGroups>identityGroups</identityGroups>
    <lastName>lastName</lastName>
    <password>password</password>
</ns3:internaluser>
```

Create User Sample Response

```
HTTP Status: 201 (Created)
Content:
N/A
```

Update User API Call

You use the Update User API call to update internal users in Cisco ISE. While updating internal users using the External RESTful Services API, if you do not intend to change the existing password, you need not supply the current password. In case you want to change the existing password when you update internal users, you can specify it in plain text.

Table 6-6 Update User Characteristics

Characteristic	Description
Description	Update the specified internal user
Synopsis	PUT /ers/config/internaluser/{internaluser-id}
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	InternalUser
Response Headers	Content-Length, Content-Type
Response Message Body	List of updated fields
Response Status	200, 400, 401, 403, 404, 415, 500

Update User Sample Request

```
Method: PUT
URI: https://10.56.13.196:9060/ers/config/internaluser/{id}
HTTP Content-Type header:
application/vnd.com.cisco.ise.identity.internaluser.1.0+xml; charset=utf-8
Request Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:internaluser name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <changePassword>true</changePassword>
    <customAttributes>
        <entry>
            <key>key2</key>
            <value>value3</value>
        </entry>
        <entry>
            <key>key1</key>
            <value>value1</value>
        </entry>
    </customAttributes>
    <email>email@domain.com</email>
    <enabled>true</enabled>
    <firstName>firstName</firstName>
    <identityGroups>identityGroups</identityGroups>
    <lastName>lastName</lastName>
    <password>password</password>
</ns3:internaluser>
```

Update User Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:updatedFields xmlns:ns2="ers.ise.cisco.com">
    <updatedField field="field1">
        <newValue>val_new</newValue>
        <oldValue>val_old</oldValue>
    </updatedField>
    <updatedField field="some other field">
        <newValue>val_new</newValue>
        <oldValue>val_old</oldValue>
    </updatedField>
</ns2:updatedFields>
```

Delete User API Call

You use the Delete User API call to delete internal users from Cisco ISE.

Table 6-7 Delete User Characteristics

Characteristic	Description
Description	Delete the specified internal user
Synopsis	DELETE /ers/config/internaluser/{internaluser-id}
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	Resource of type InternalUser
Response Status	204, 400, 401, 403, 404, 415, 500

Delete User Sample Request

```
Method: DELETE
URI: https://10.56.13.196:9060/ers/config/internaluser/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.identity.internaluser.1.0+xml
Request Content:
N/A
```

Delete User Sample Response

```
HTTP Status: 204 (No Content)
Content:
N/A
```

Endpoint External RESTful Services API Calls

The internal user ID used in these API calls is the UUID type stored in the Cisco ISE database.

Note These examples are not meant to be used as is because they have references to DB data. You should treat it as a basic template and edit it before sending to server.

Table 6-8 External RESTful Services API Calls Available for Endpoints

Operation	Method	URL	Content	QueryString
Get All Endpoints	GET	/ers/config/endpoint	—	page, size, sortacs or sortdsn, filter
Get Endpoint	GET	/ers/config/endpoint/{endpoint-id}	—	—
Create Endpoint	POST	/ers/config/endpoint/	endpoint	—
Update Endpoint	PUT	/ers/config/endpoint/{endpoint-id}	endpoint	—
Delete Endpoint	DELETE	/ers/config/endpoint/{endpoint-id}	—	—
Register Endpoint	PUT	/ers/config/endpoint/register	endpoint	—
Deregister Endpoint	PUT	/ers/config/endpoint/{endpoint-id}/deregister	—	—

Endpoint Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpoint name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <groupId>groupId</groupId>
    <identityStore>identityStore</identityStore>
    <identityStoreId>identityStoreId</identityStoreId>
    <mac>00:01:02:03:04:05</mac>
    <portalUser>portalUser</portalUser>
    <profileId>profileId</profileId>
    <staticGroupAssignment>true</staticGroupAssignment>
    <staticProfileAssignment>false</staticProfileAssignment>
</ns3:endpoint>
```

Get All Endpoints API Call

You use the Get All Endpoints API call to retrieve a list of endpoints.

Table 6-9 Get All Endpoints Characteristics

Characteristic	Description
Description	Retrieve collection of endpoints
Synopsis	GET ers/config/endpoint
Request Headers	Accept, Authorization, Host
QueryString	start, size, sortbyacn, sortbydcn, filter (fields supported by filter: mac, portalUser, profileId, staticGroupAssignment, staticProfileAssignment)
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	SearchResult
Response Status	200, 400, 401, 403, 404, 415, 500

Get All Endpoints Sample Request

```
Method: GET
URI: https://10.56.13.196:9060/ers/config/endpoint
HTTP Accept header:
application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
Request Content:
N/A
```

Get All Endpoints Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
    <nextPage type="application/xml" href="link-to-next-page" rel="next"/>
    <previousPage type="application/xml" href="link-to-previous-page" rel="previous"/>
    <resources>
        <resource name="name1" id="id1" description="description1"/>
        <resource name="name2" id="id2" description="description2"/>
    </resources>
</ns2:searchResult>
```

Note The endpoint do not have hyperlinks for the complete object as it is not supported. Only the name description and ID are presented.

Get Endpoint API Call

You use Get Endpoint API call to retrieve an endpoint using the user ID.

Table 6-10 Get Endpoint Characteristics

Characteristics	Description
Description	Retrieve the specified endpoint
Synopsis	GET /ers/config/endpoint/{endpoint-id}
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	Resource of type InternalUser
Response Status	200, 400, 401,403, 404, 415, 500

Get Endpoint Sample Request

```
Method: GET
URI: https://10.56.13.196:9060/ers/config/endpoint/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
Request Content:
N/A
```

Get Endpoint Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpoint name="name" id="id" description="description" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="identity.ers.ise.cisco.com">
    <groupId>groupId</groupId>
    <identityStore>identityStore</identityStore>
    <identityStoreId>identityStoreId</identityStoreId>
    <mac>00:01:02:03:04:05</mac>
    <portalUser>portalUser</portalUser>
    <profileId>profileId</profileId>
    <staticGroupAssignment>true</staticGroupAssignment>
    <staticProfileAssignment>false</staticProfileAssignment>
</ns3:endpoint>
```

Create Endpoint API Call

You use the Create Endpoint API call to create endpoints in Cisco ISE.

Table 6-11 Create Endpoint Characteristics

Characteristic	Description
Description	Create the specified endpoint
Synopsis	POST /ers/config/endpoint
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	Endpoint
Response Headers	Content-Length, Content-Type
Response Message Body	Resource of type InternalUser
Response Status	200, 400, 401,403, 404, 415, 500

Create Endpoint Sample Request

```
Method: POST
URI: https://10.56.13.196:9060/ers/config/endpoint
HTTP Content-Type header:
application/vnd.com.cisco.ise.identity.endpoint.1.0+xml; charset=utf-8
Request Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpoint name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
<groupId>groupId</groupId>
    <identityStore>identityStore</identityStore>
    <identityStoreId>identityStoreId</identityStoreId>
    <mac>00:01:02:03:04:05</mac>
    <portalUser>portalUser</portalUser>
    <profileId>profileId</profileId>
    <staticGroupAssignment>true</staticGroupAssignment>
    <staticProfileAssignment>false</staticProfileAssignment>
</ns3:endpoint>
```

Create Endpoint Sample Response

```
HTTP Status: 201 (Created)
Content:
N/A
```

Update Endpoint API Call

You use the Update Endpoint API call to update endpoints in Cisco ISE.

Table 6-12 Update Endpoint Characteristics

Characteristic	Description
Description	Update the specified endpoint
Synopsis	PUT /ers/config/endpoint/{endpoint-id}
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	Endpoint
Response Headers	Content-Length, Content-Type
Response Message Body	List of updated fields
Response Status	200, 400, 401,403, 404, 415, 500

Update Endpoint Sample Request

```
Method: PUT
URI: https://10.56.13.196:9060/ers/config/endpoint/{id}
HTTP Content-Type header:
application/vnd.com.cisco.ise.identity.endpoint.1.0+xml; charset=utf-8
Request Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpoint name="name" id="id" description="description" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="identity.ers.ise.cisco.com">
    <groupId>groupId</groupId>
    <identityStore>identityStore</identityStore>
    <identityStoreId>identityStoreId</identityStoreId>
    <mac>00:01:02:03:04:05</mac>
    <portalUser>portalUser</portalUser>
    <profileId>profileId</profileId>
    <staticGroupAssignment>true</staticGroupAssignment>
    <staticProfileAssignment>false</staticProfileAssignment>
</ns3:endpoint>
```

Update Endpoint Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:updatedFields xmlns:ns2="ers.ise.cisco.com">
    <updatedField field="field1">
        <newValue>val_new</newValue>
        <oldValue>val_old</oldValue>
    </updatedField>
    <updatedField field="some other field">
        <newValue>val_new</newValue>
        <oldValue>val_old</oldValue>
    </updatedField>
</ns2:updatedFields>
```

Delete Endpoint API Call

You use the Delete Endpoint API call to delete endpoints in Cisco ISE.

Table 6-13 Delete Endpoint Characteristics

Characteristic	Description
Description	Delete the specified endpoint
Synopsis	DELETE /ers/config/endpoint/{endpoint-id}
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	—
Response Status	200, 400, 401,403, 404, 415, 500

Delete Endpoint Sample Request

```
Method: DELETE
URI: https://10.56.13.196:9060/ers/config/endpoint/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.identity.endpoint.1.0+xml
Request Content:
N/A
```

Delete Endpoint Sample Response

```
HTTP Status: 204 (No Content)
Content:
N/A
```

Register Endpoint API Call

You use the Register Endpoint API call to register endpoints in Cisco ISE. If the endpoint already exists, it will be registered. If it does not exist, it will be created and then registered. In both scenarios, the return status will be 204, which means no content. Similar to the GUI registration flow, the endpoint is statically assigned to the Registered Devices group and the portal user and identity store will be set as specified in the content.

Table 6-14 Register Endpoint Characteristics

Characteristic	Description
Description	Register the specified endpoint
Synopsis	PUT /ers/config/endpoint/register
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	endpoint

Table 6-14 Register Endpoint Characteristics (continued)

Characteristic	Description
Response Headers	Content-Length, Content-Type
Response Message Body	List of updated fields
Response Status	202, 400, 401, 403, 404, 415, 500

Register Endpoint Sample Request

```
Method: PUT
URI: https://10.56.13.196:9060/ers/config/endpoint/register
HTTP Content-Type header:
application/vnd.com.cisco.ise.identity.endpoint.1.0+xml; charset=utf-8
Request Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpoint name="name" id="id" description="description" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="identity.ers.ise.cisco.com">
    <groupId>groupId</groupId>
    <identityStore>identityStore</identityStore>
    <identityStoreId>identityStoreId</identityStoreId>
    <mac>00:01:02:03:04:05</mac>
    <portalUser>portalUser</portalUser>
    <profileId>profileId</profileId>
    <staticGroupAssignment>true</staticGroupAssignment>
    <staticProfileAssignment>false</staticProfileAssignment>
</ns3:endpoint>
```

Register Endpoint Sample Response

```
HTTP Status: null
Content:
N/A
```

Deregister Endpoint API Call

You use the Deregister Endpoint API call to deregister endpoints in Cisco ISE.

Table 6-15 Deregister Endpoint Characteristics

Characteristic	Description
Description	Deregister the specified endpoint
Synopsis	PUT /ers/config/endpoint/{endpoint-id}/deregister
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	—

Table 6-15 Deregister Endpoint Characteristics (continued)

Characteristic	Description
Response Message Body	—
Response Status	202, 400, 401, 403, 404, 415, 500

Deregister Endpoint Sample Request

```
Method: PUT
URI: https://10.56.13.196:9060/ers/config/endpoint/{id}/deregister
HTTP Content-Type header:
application/vnd.com.cisco.ise.identity.endpoint.1.0+xml; charset=utf-8
Request Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpoint name="name" id="id" description="description" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="identity.ers.ise.cisco.com">
    <groupId>groupId</groupId>
    <identityStore>identityStore</identityStore>
    <identityStoreId>identityStoreId</identityStoreId>
    <mac>00:01:02:03:04:05</mac>
    <portalUser>portalUser</portalUser>
    <profileId>profileId</profileId>
    <staticGroupAssignment>true</staticGroupAssignment>
    <staticProfileAssignment>false</staticProfileAssignment>
</ns3:endpoint>
```

Deregister Endpoint Sample Response

```
HTTP Status: null
Content:
N/A
```

Endpoint Identity Group External RESTful Services API Calls

The internal user ID used in these API calls is the UUID type stored in the Cisco ISE database.

Note The examples shown in the subsequent sections are not meant to be used as is because they have references to DB data. You should treat it as a basic template and edit it before sending to server.

Table 6-16 External RESTful Services API Calls Available for Endpoint Identity Groups

Operation	Method	URL	Content	QueryString
Get All Endpoint Groups	GET	/ers/config/endpointgroup	—	page, size, sortacs or sortdsn, filter
Get Endpoint Group	GET	/ers/config/endpointgroup/{endpointgroup-id}	—	—
Create Endpoint Group	POST	/ers/config/endpointgroup/	Endpointgroup	—
Update Endpoint Group	PUT	/ers/config/endpointgroup/{endpointgroup-id}	Endpointgroup	—
Delete Endpoint Group	DELETE	/ers/config/endpointgroup/{endpointgroup-id}	—	—

Endpoint Identity Groups Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpointgroup name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <systemDefined>true</systemDefined>
</ns3:endpointgroup>
```

Get All Endpoint Identity Groups API Call

You use the Get All Endpoint Identity Groups API call to retrieve a collection of endpoint identity groups.

Table 6-17 Get All Endpoint Identity Groups Characteristics

Characteristic	Description
Description	Retrieve a collection of endpoint groups
Synopsis	GET /ers/config/endpointgroup
Request Headers	Accept, Authorization, Host
QueryString	start, size, sortbyacn, sortbydcn, filter
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	SearchResult
Response Status	202, 400, 401, 403, 404, 415, 500

Get All Endpoint Identity Groups Sample Request

```
Method: GET
URI: https://10.56.13.196:9060/ers/config/endpointgroup
HTTP Accept header:
application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xml
```

```
Request Content:
N/A
```

Get All Endpoint Identity Groups Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
    <nextPage type="application/xml" href="link-to-next-page" rel="next"/>
    <previousPage type="application/xml" href="link-to-previous-page" rel="previous"/>
    <resources>
        <resource name="name1" id="id1" description="description1"/>
        <resource name="name2" id="id2" description="description2"/>
    </resources>
</ns2:searchResult>
```

Note The endpoint do not have hyperlinks for the complete object, only the name, description, and ID are presented.

Get Endpoint Identity Group API Call

You use the Get Endpoint Identity Group API call to retrieve the specified endpoint identity group.

Table 6-18 Get Endpoint Identity Group Characteristics

Characteristic	Description
Description	Retrieve the specified endpoint group
Synopsis	GET /ers/config/endpoint/{endpointgroup-id}
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	Resource of type Endpoint
Response Status	200, 400, 401,403, 404, 415, 500

Get Endpoint Identity Group Sample Request

```
Method: GET
URI: https://10.56.13.196:9060/ers/config/endpointgroup/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xml
Request Content:
N/A
```

Get Endpoint Identity Group Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpointgroup name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <systemDefined>true</systemDefined>
</ns3:endpointgroup>
}
```

Create Endpoint Identity Group API Call

You use the Create Endpoint Identity Group API call to create a specified endpoint identity group.

:

Table 6-19 Create Endpoint Identity Group Characteristics

Characteristic	Description
Description	Create the specified endpoint group
Synopsis	POST /ers/config/endpoint/
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	Endpoint Group
Response Headers	Content-Length, Content-Type, Location
Response Message Body	Resource of type Endpoint Group
Response Status	201, 400, 401, 403, 415, 500

Create Endpoint Identity Group Sample Request

```
Method: POST
URI: https://10.56.13.196:9060/ers/config/endpointgroup
HTTP Content-Type header:
application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xml; charset=utf-8
Request Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpointgroup name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <systemDefined>true</systemDefined>
</ns3:endpointgroup>
```

Create Endpoint Identity Group Sample Response

```
HTTP Status: 201 (Created)
Content:
N/A
```

Update Endpoint Identity Group API Call

You use the Update Endpoint Identity Group API call to update a specified endpoint group.

Table 6-20 Update Endpoint Identity Group Characteristics

Characteristic	Description
Description	Update the specified endpoint group
Synopsis	PUT /ers/config/endpoint/{endpointgroup-id}
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	Endpoint Group
Response Headers	Content-Length, Content-Type
Response Message Body	List of updated fields
Response Status	200, 400, 401, 403, 404, 415, 500

Update Endpoint Identity Group Sample Request

```
Method: PUT
URI: https://10.56.13.196:9060/ers/config/endpointgroup/{id}
HTTP Content-Type header:
application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xml; charset=utf-8
Request Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:endpointgroup name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <systemDefined>true</systemDefined>
</ns3:endpointgroup>
```

Update Endpoint Identity Group Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:updatedFields xmlns:ns2="ers.ise.cisco.com">
    <updatedField field="field1">
        <newValue>val_new</newValue>
        <oldValue>val_old</oldValue>
    </updatedField>
    <updatedField field="some other field">
        <newValue>val_new</newValue>
        <oldValue>val_old</oldValue>
    </updatedField>
</ns2:updatedFields>
}
```

Delete Endpoint Identity Group API Call

You use the Delete Endpoint Identity Group API call to delete a specified endpoint identity group.

Table 6-21 Delete Endpoint Identity Group Characteristics

Characteristic	Description
Description	Delete the specified endpoint group
Synopsis	DELETE /ers/config/endpointgroup/{endpointgroup-id}
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	—
Response Status	204, 400, 401, 403, 404, 415, 500

Delete Endpoint Identity Group Sample Request

```
Method: DELETE
URI: https://10.56.13.196:9060/ers/config/endpointgroup/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.identity.endpointgroup.1.0+xml
Request Content:
N/A
```

Delete Endpoint Identity Group Sample Response

```
HTTP Status: 204 (No Content)
Content:
N/A
```

Profiler Profile External RESTful Services API Calls

Profiler Profile API calls enable you to search profiles. Profiler profile is the profile the endpoint is classified as or statically assigned to. Profiler Profile is also called profiler policy. The profile Id is an attribute on the endpoint. Endpoints can be filtered by names.

Note The examples shown in the following sections are not meant to be used as is because they have references to DB data. You should treat it as a basic template and edit it before sending to server.

Table 6-22 External RESTful Services API Calls Available for Profiler Profile

Operation	Method	URL	Content	QueryString
Get All Profiles	GET	/ers/config/profilerprofile	—	Start Size sortacs or sortdsn filter
Get Profile (Get by ID)	GET	/ers/config/profilerprofile/{id}	—	—
Create Profile	POST	ers/config/profilerprofile	—	—

Profiler Profile Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:profilerprofile name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com"/>
```

Get All Profiles API Call

You use the Get All Profiles API call to retrieve a list of Profiler profiles.

Table 6-23 Get All Profiles Characteristics

Characteristic	Description
Description	Retrieves a collection of profiler profiles
Synopsis	GET /ers/config/profilerprofile
Request Headers	Accept, Authorization, Host
QueryString	start, size, sortbyacn, sortbydcn, filter (fields supported by filter: name)
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	SearchResult
Response Status	200, 400, 401, 403, 404, 415, 500

Get All Profiles Sample Request

```
Method: GET
URI: https://10.56.13.196:9060/ers/config/profilerprofile
HTTP Accept header:
application/vnd.com.cisco.ise.identity.profilerprofile.1.0+xml
Request Content:
N/A
```

Get All Profiles Sample Response

```
HTTP Status: 200 (OK)

Content:



<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">

    <nextPage type="application/xml" href="link-to-next-page" rel="next"/>

    <previousPage type="application/xml" href="link-to-previous-page" rel="previous"/>

    <resources>

        <resource name="name1" id="id1" description="description1"/>

        <resource name="name2" id="id2" description="description2"/>

    </resources>

</ns2:searchResult>
```

Note The Profiler profile will not have the hyperlinks for the complete object as it is not supported in this release. And only name description and id will be presented.

Get Profile API Call

You use the Get Profile API call to retrieve a specific profiler profile.

Table 6-24 Get Profile Characteristics

Characteristic	Description
Description	Retrieves the specified profiler profile.
Synopsis	GET ers/config/profilerprofile/{id}
Request Headers	Accept, Authorization, Host
QueryString	start, size, sortbyacn, sortbydcn, filter (fields supported by filter: name)
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	SearchResult
Response Status	200, 400, 401, 403, 404, 415, 500

Get Profile Sample Request

```
Method: GET

URI: https://10.56.13.196:9060/ers/config/profilerprofile/{id}

HTTP Accept header:
```

```
application/vnd.com.cisco.ise.identity.profilerprofile.1.0+xml

Request Content:

N/A
```

Get Profile Sample Response

```
HTTP Status: 200 (OK)

Content:



<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ns3:profilerprofile name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com"/>
```

Create Profiler Profile API Call

You use the Create Profile API call to create a profiler profile.

Table 6-25 Create Profile Characteristics

Characteristic	Description
Description	Create a profiler profile
Synopsis	POST ers/config/profilerprofile
Request Headers	Accept, Authorization, Host
QueryString	start, size, sortbyacn, sortbydcn, filter (fields supported by filter: name)
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	—
Response Status	200, 400, 401, 403, 404, 415, 500

Create Profile Sample Request

```
Method: POST

URI: https://10.56.13.196:9060/ers/config/profilerprofile

HTTP Content-Type header:

application/vnd.com.cisco.ise.identity.profilerprofile.1.0+xml; charset=utf-8

Request Content:
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ns3:profilerprofile name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com"/>
```

Create Profile Sample Response

```
HTTP Status: 201 (Created)

Content:

N/A
```

Identity Group External RESTful Services API Calls

External RESTful Services Identity Groups API calls enable you to search Identity Groups.

Note The examples shown in the subsequent sections are not meant to be used as is because they have references to DB data. You should treat it as a basic template and edit it before sending to server.

Table 6-26 External RESTful Services API Calls Available for Identity Groups

Operation	Method	URL	Content	QueryString
Get Identity Group (Get by ID)	GET	/ers/config/identitygroup /{id}	—	—
List Identity Groups	GET	/ers/config/identitygroup	—	page, size, sortacs or sortdsn, filter
Get IdentityGroup Resource Version Info	GET	/ers/config/identitygroup /version	—	—

Identity Group Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ns3:identitygroup name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">

    <parent>parent</parent>

</ns3:identitygroup>
```

Get All Identity Groups API Call

You use the Get All Identity Groups API call to retrieve identity groups by ID in Cisco ISE.

Table 6-27 Get All Identity Groups Characteristics

Characteristic	Description
Description	Retrieve identity group resources by ID.
Synopsis	GET /ers/config/identitygroup/{id}
Request Headers	Accept, Authorization, Host
QueryString	start, size, sortbyacn, sortbydcn, filter (fields supported by filter: name, description)
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	—
Response Status	200, 400, 401, 403, 404, 415, 500

Get All Identity Groups Sample Request

```
Method: GET
URI: https://10.56.13.196:9060/ers/config/identitygroup/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.identity.identitygroup.1.0+xml
Request Content:
N/A
```

Get All Identity Groups Sample Response

```
HTTP Status: 200 (OK)

Content:



<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ns3:identitygroup name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">

    <parent>parent</parent>

</ns3:identitygroup>
```

Get Identity Group API Call

You use the Get Identity Group API call to retrieve a specific identity group in Cisco ISE.

Table 6-28 Get Identity Group Characteristics

Characteristic	Description
Description	Retrieves a specified identity group resource
Synopsis	GET /ers/config/identitygroup

Table 6-28 Get Identity Group Characteristics (continued)

Characteristic	Description
Request Headers	Accept, Authorization, Host
QueryString	start, size, sortbyacn, sortbydcn, filter (fields supported by filter: name, description)
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	SearchResult
Response Status	200, 400, 401, 403, 404, 415, 500

Get Identity Group Sample Request

```
Method: GET
URI: https://10.56.13.196:9060/ers/config/identitygroup/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.identity.identitygroup.1.0+xml
Request Content:
N/A
```

Get Identity Group Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:identitygroup name="name" id="id" description="description"
xmlns:ns2="ers.ise.cisco.com" xmlns:ns3="identity.ers.ise.cisco.com">
    <parent>parent</parent>
</ns3:identitygroup>
```

External RESTful Services API Calls for Security Group Tags

Security Group Tags (SGT) API calls enable you to search SGTs. The internal user ID used in these API calls is the UUID type stored in the Cisco ISE database.

Note The examples shown in the subsequent sections are not meant to be used as is because they have references to DB data. You should treat it as a basic template and edit it before sending to server.

Table 6-29 External RESTful Services API Calls Available for SGTs

Operation	Method	URL	Content	QueryString
Get All SGTs	GET	/ers/config/sgt	—	page, size, sortacs or sortdsn, filter
Get SGT (Get by ID)	GET	/ers/config/sgt/{sgt-id}	—	—

Get All SGTs API Call

You use the Get All SGTs API call to retrieve a collection of SGT resources.

Table 6-30 Get All SGTs Characteristics

Characteristic	Description
Description	Retrieve a collection of SGT resources
Synopsis	GET /ers/config/sgt
Request Headers	Accept, Authorization, Host
QueryString	start, size, sortbyacn, sortbydcn, filter (fields supported by filter: name)
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	SearchResult
Response Status	200, 400, 401, 403, 404, 415, 500

Get All SGTs Sample Request

```
Method: GET
URI: https://10.56.13.196:9060/ers/config/sgt
HTTP Accept header:
application/vnd.com.cisco.ise.sga.sgt.1.0+xml
Request Content:
N/A
```

Get All SGTs Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:searchResult total="2" xmlns:ns2="ers.ise.cisco.com">
    <nextPage type="application/xml" href="link-to-next-page" rel="next"/>
    <previousPage type="application/xml" href="link-to-previous-page" rel="previous"/>
    <resources>
        <resource name="name1" id="id1" description="description1"/>
        <resource name="name2" id="id2" description="description2"/>
    </resources>
</ns2:searchResult>
```

Get SGT API Call

You use the Get SGT API call to retrieve a specific SGT resource.

Table 6-31 Get SGT Characteristics

Characteristic	Description
Description	Retrieve the specified SGT
Synopsis	GET /ers/config/sgt/{sgt-id}
Request Headers	Accept, Authorization, Host
QueryString	—
Request Message Body	—
Response Headers	Content-Length, Content-Type
Response Message Body	Resource of type InternalUser
Response Status	200, 400, 401, 403, 404, 415, 500

Get SGT Sample Request

```
Method: GET
URI: https://10.56.13.196:9060/ers/config/sgt/{id}
HTTP Accept header:
application/vnd.com.cisco.ise.sga.sgt.1.0+xml
Request Content:
N/A
```

Get SGT Sample Response

```
HTTP Status: 200 (OK)
Content:


<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:sgt name="name" id="id" description="description" xmlns:ns2="ers.ise.cisco.com"
xmlns:ns3="sga.ers.ise.cisco.com">
    <generationId>generationId</generationId>
    <isTagFromRange>isTagFromRange</isTagFromRange>
    <value>1</value>
</ns3:sgt>
}
```

Using a REST API Client

To build and test applications using the External RESTful Services API, you can use any industry-standard REST API client, such as the POSTMAN plugin for Google Chrome.

Designed according to REST architecture and principles, the POSTMAN Chrome plugin is an easy to use REST client. It allows you to save collections of requests and provides many features that are useful while working in a browser environment and have lightweight specific tasks. You can also use the POStMAN plugin for testing the External RESTful Services APIs.

POSTMAN enables you to send and retrieve standard HTTP and HTTPS requests and responses using the Google Chrome web browser. You can use the following standard HTTP methods to perform CRUD operations on Cisco ISE resources:

- GET
- POST
- PUT
- DELETE

The External RESTful Services API enables you to use these HTTP requests in various API calls, which in turn enable you to perform operations on Cisco ISE servers. For a comprehensive list of operations in which these HTTP requests are used, see External RESTful Services Calls, page 6-1.

Note To download the POSTMAN plugin, go to https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm?hl=en. For more information on using the POSTMAN plugin, go to https://github.com/a85/POSTMan-Chrome-Extension/wiki.

Making an External RESTful Services API Call Using GET

The GET method requests a representation of the specified resource. Requests using GET only retrieve data and do not have any other effect.

You can perform a search operation by sending a GET request to a resource URI. By default, the result is the first page (page index = 0) with default size of 20. By adding a filter, Sort, and paging parameters to the URI, you can control the results of the search operation. The search returns the following result type:

```
SearchResult: MediaType[MediaType:
[application/vnd.com.cisco.ise.ers.searchresult.1.0+xml]
```

The resulting resources are a collection of base resources that contain the name, id, description, and a link to the full representation of the resource.

Note An External RESTful Services API call uses the GET HTTP method in addition to other components of the External RESTful Services API, which are not described in this section. For more details on various External RESTful Services API components, such as the characteristics, requests, and responses, see External RESTful Services Calls.

The request body of the External RESTful Services API call that uses the GET HTTPS method contains the following three building blocks:

- URI
- Accept Header
- Authorization Header

URI

The GET method sends the Uniform Resource Identifier (URI) to the Cisco ISE server and the HTTP reply is the raw result data. A typical URI must adhere to the following format:

- *https://<Cisco ISE Server address:<port>/<namespace>/config/<Cisco ISE Resource Name>*

  Where <*Cisco ISE Server Address*> denotes the server address of the Cisco ISE server, <*port*> denotes port 9060, <*namespace*> denotes the namespace to which the Cisco ISE resource belongs to, and <*Cisco ISE Resource Name*> denotes the name of the Cisco ISE resource.

The following example shows a URI that requests data for the interaluser Cisco ISE resource:

- *https://10.56.13.196:9060/ers/config/internaluser.*

Note The URI is not the request body; it is just a URL. This URL is sent to the server using the GET method.

Accept Header

The Accept header must adhere to the following format:

- *application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major version>.<minor version>+xml*

Where <*resource-namespace*> denotes the namespace to which the Cisco ISE resource belongs to, <*resource-type*> denotes the type of Cisco ISE resource, <*major-version*> denotes the major version number of the Cisco ISE deployment, and <*minor-version*> denotes the minor version number of the deployment.

The following example shows a typical Accept header:

- *application/vnd.com.cisco.ise.identity.internaluser.1.0+xml*

Authorization Header

The Authorization header contains the encryption authorization key that is embedded in the GET request. After specifying the authorization credentials, you must generate the encryption key, which is then embedded in the request body.

Note For more information on generating an encryption key, see Making a GET Request Using POSTMAN, page 6-31.

Making a GET Request Using POSTMAN

Step 1 Open the POSTMAN plugin in the Google Chrome browser.

Step 2 Create a new collection using the options in the left pane.

Note For more information on using the POSTMAN plugin, go to https://github.com/a85/POSTMan-Chrome-Extension/wiki.

Step 3 From the drop-down menu, choose **GET**.

Step 4 In the URL address bar, enter a URI.

The URI specifies the Cisco ISE server with which you are trying to communicate and the Cisco ISE resource that you are trying to access. For more information on the format of the URI, see URI, page 6-30.

Step 5 Click the **Basic Auth** tab.

The options that enable you to specify the user access credentials appear.

Step 6 Specify the access credentials in the Username and Password fields and click **Refresh Headers**.

POSTMAN displays an Authorization header with an encryption key.

Step 7 Add an Accept header by specifying the following value:
application/vnd.com.cisco.ise.ers.<namespace>.<ise resource>.1.0+xml

Note For more information on the Accept header, see Accept Header, page 6-31.

Step 8 Click **Send**.

The POSTMAN plugin displays a 200 OK status response indicating that the request is successful. The request also returns the details of the resources that you have specified in the URI.

Making an External RESTful Services API Call Using POST

The POST method requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI.

You can create a resource by sending a POST request to a resource URI. The request content holds XML representation of the resource, and must be well formatted according to the schema. The request header holds the encrypted authorization and the content-type that defines the resource content and its version.

Note An External RESTful Services API call uses the POST HTTP method in addition to other components of the External RESTful Services API, which are not described in this section. For more details on various components, such as characteristics, requests, and responses, see External RESTful Services Calls.

The request body of the External RESTful Services API call that uses the POST HTTP method contains the following three building blocks:

- URI
- Content-Type Header
- Authorization Header

URI

The POST method sends the Uniform Resource Identifier (URI) to the Cisco ISE server. A typical URI must adhere to the following format:

- *https://<Cisco ISE Server address:<port>/<namespace>/config/<Cisco ISE Resource Name>*

  Where *<Cisco ISE Server Address>* denotes the server address of the Cisco ISE server, *<port>* denotes port 9060, *<namespace>* denotes the namespace to which the Cisco ISE resource belongs to, and *<Cisco ISE Resource Name>* denotes the name of the Cisco ISE resource.

The following example shows a URI that requests data for the interaluser Cisco ISE resource:

- *https://10.56.13.196:9060/ers/config/internaluser.*

Note The URI is not the request body; it is just a URL. This URL is sent to the server using the POST method.

Content-Type Header

The Content-Type header must adhere to the following format:

- *application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major version>.<minor version>+xml*

  Where <*resource-namespace*> denotes the namespace to which the Cisco ISE resource belongs to, <*resource-type*> denotes the type of Cisco ISE resource, <*major-version*> denotes the major version number of the Cisco ISE deployment, and <*minor-version*> denotes the minor version number of the deployment.

The following example shows a typical Content-Type header:

- *application/vnd.com.cisco.ise.identity.internaluser.1.0+xml*

Authorization Header

The Authorization header contains the encryption authorization key that is embedded in the POST request. After specifying the authorization credentials, you must generate the encryption key, which is then embedded in the request body.

Note For more information on generating an encryption key, see Making a POST Request Using POSTMAN, page 6-33.

Making a POST Request Using POSTMAN

Step 1 Open the POSTMAN plugin in the Google Chrome browser.

Step 2 Create a new collection using the options in the left pane.

Note For more information on using the POSTMAN plugin, go to https://github.com/a85/POSTMan-Chrome-Extension/wiki.

Step 3 From the drop-down menu, choose **POST**.

Step 4 In the URL address bar, enter the URI.

The URI specifies the Cisco ISE server with which you are trying to communicate and the Cisco ISE resource that you are trying to access. For more information on the format of the URI, see URI, page 6-32.

Step 5 Click the **Basic Auth** tab.

The options that enable you to specify the user access credentials appear.

Step 6 Specify the access credentials in the Username and Password fields and click **Refresh Headers**.

POSTMAN displays an Authorization header with an encryption key.

Step 7 Add a Content-Type header by specifying the following value:
application/vnd.com.cisco.ise.ers.<namespace>.<ise resource>.1.0+xml

Note For more information on the Content-Type header, see Content-Type Header, page 6-33.

Step 8 From the drop-down menu that appears next to the **raw** button, choose **XML**.

Step 9 Click **raw**.

Step 10 The POSTMAN plugin opens an editing pane that enables you to specify the body of the POST request.

Step 11 Enter the message body of your POST request in the editing pane.

Note The message body must contain the details corresponding to the Cisco ISE resource that you trying to create on the Cisco ISE server. For example, while creating an internal user, you must specify details such as the name and description of the internal user, password, and so on. For more details on the message body of the External RESTful Services API calls that use the POST request and the details of the Cisco ISE resources that you need to specify, see External RESTful Services Calls.

Step 12 Click **Send**.

The POSTMAN plugin displays a 201 CREATED status response indicating that the request is successful. You can log on to Cisco ISE to verify whether the resource you added appears in the GUI.

Making an External RESTful Services API Call Using PUT

The PUT method requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified; if the URI does not point to an existing resource, then the server can create the resource with that URI.

You can update a resource by sending a PUT request to a resource URI. The request content holds XML representation of the resource, and must be well formatted according the schema. The request header holds the encrypted authorization and the content-type that defines the resource content and its version.

Note An External RESTful Services API call uses the PUT HTTP method in addition to other components of the External RESTful Services API, which are not described in this section. For more details on various components, such as characteristics, requests, and responses, see External RESTful Services Calls.

The request body of the External RESTful Services API call that uses the PUT HTTP method contains the following three building blocks:

- URI
- Content-Type Header
- Authorization Header

URI

The PUT method sends the Uniform Resource Identifier (URI) to the Cisco ISE server. A typical URI must adhere to the following format:

- *https://<Cisco ISE Server address:<port>/<namespace>/config/<Cisco ISE Resource Name>*

  Where <*Cisco ISE Server Address*> denotes the server address of the Cisco ISE server, <*port*> denotes the port 9060, <*namespace*> denotes the namespace to which Cisco ISE resource belongs to, and <*Cisco ISE Resource Name*> denotes the name of the Cisco ISE resource.

The following example shows a URI that requests data for the interaluser Cisco ISE resource:

- *https://10.56.13.196:9060/ers/config/internaluser.*

Note The URI is not the request body; it is just a URL. This URL is sent to the server using the PUT method.

Content-Type Header

The Content-Type header must adhere to the following format:

- *application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major version>.<minor version>+xml*

  Where <*resource-namespace*> denotes the namespace to which the Cisco ISE resource belongs to, <*resource-type*> denotes the type of Cisco ISE resource, <*major-version*> denotes the major version number of the Cisco ISE deployment, and <*minor-version*> denotes the minor version number of the deployment.

The following example shows a typical Content-Type header:

- *application/vnd.com.cisco.ise.identity.internaluser.1.0+xml*

Authorization Header

The Authorization header contains the encryption authorization key that is embedded in the PUT request. After specifying the authorization credentials, you must generate the encryption key, which is then embedded in the request body.

Note For more information on generating an encryption key, see Making a PUT Request Using POSTMAN, page 6-35.

Making a PUT Request Using POSTMAN

Step 1 Open the POSTMAN plugin in the Google Chrome browser.

Step 2 Create a new collection using the options in the left pane.

Note For more information on using the POSTMAN plugin, go to https://github.com/a85/POSTMan-Chrome-Extension/wiki.

Step 3 From the drop-down menu, choose **PUT**.

Step 4 In the URL address bar, enter the URI.

The URI specifies the Cisco ISE server with which you are trying to communicate and the Cisco ISE resource that you are trying to access. For more information on the format of the URI, see URI, page 6-34.

Step 5 Click the **Basic Auth** tab.

The options that enable you to specify the user access credentials appear.

Step 6 Specify the access credentials in the Username and Password fields and click **Refresh Headers**.

POSTMAN displays an Authorization header with an encryption key.

Step 7 Add a Content-Type header by specifying the following value:
application/vnd.com.cisco.ise.ers.<namespace>.<ise resource>.1.0+xml

Note For more information on the Content-Type header, see Content-Type Header, page 6-35.

Step 8 From the drop-down menu that appears next to the **raw** button, choose **XML**.

Step 9 Click **raw**.

Step 10 The POSTMAN plugin opens an editing pane that enables you to specify the body of the POST request.

Step 11 Enter the message body of your POST request in the editing pane.

Note The message body must contain the details corresponding to the Cisco ISE resource that you trying to update on the Cisco ISE server. For example, while updating an internal user, you must specify details such as the name and description of the internal user, password, and so on. For more details on the message body of the External RESTful Services API calls that use the PUT request and the details of the Cisco ISE resources that you need to specify, see External RESTful Services Calls.

Step 12 Click **Send**.

The POSTMAN plugin displays a 201 CREATED status response indicating that the request is successful. You can log on to Cisco ISE to verify whether the resource you added appears in the GUI.

Making an External RESTful Services API Call Using DELETE

The DELETE method deletes the specified resource.

You can delete a resource by sending a DELETE request to a resource URI. The request header holds the encrypted authorization and the content-type that defines the resource content and its version.

Note An External RESTful Services API call uses the DELETE HTTP method in addition to other components of the External RESTful Services API, which are not described in this section. For more details on various components, such as characteristics, requests, and responses, see External RESTful Services Calls.

The request body of the External RESTful Services API call that uses the DELETE HTTP method contains the following three building blocks:

- URI
- Accept Header
- Authorization Header

URI

The DELETE method sends the Uniform Resource Identifier (URI) to the Cisco ISE server. A typical URI must adhere to the following format:

- *https://<Cisco ISE Server address:<port>/<namespace>/config/<Cisco ISE Resource Name>*

  Where <*Cisco ISE Server Address*> denotes the server address of the Cisco ISE server, <*port*> denotes the port 9060, <*namespace*> denotes the namespace to which the Cisco ISE resource belongs to, and <*Cisco ISE Resource Name*> denotes the name of the Cisco ISE resource.

The following example shows a URI that requests data for the interaluser Cisco ISE resource:

- *https://10.56.13.196:9060/ers/config/internaluser.*

Note The URI is not the request body; it is just a URL. This URL is sent to the server using the GET method.

Accept Header

The Accept header must adhere to the following format:

- *application/vnd.com.cisco.ise.<resource-namespace>.<resource-type>.<major version>.<minor version>+xml*

Where <*resource-namespace*> denotes the namespace to which the Cisco ISE resource belongs to, <*resource-type*> denotes the type of Cisco ISE resource, <*major-version*> denotes the major version number of the Cisco ISE deployment, and <*minor-version*> denotes the minor version number of the deployment.

The following example shows a typical accept header:

- *application/vnd.com.cisco.ise.identity.internaluser.1.0+xml*

Authorization Header

The Authorization header contains the encryption authorization key that is embedded in the DELETE request. After specifying the authorization credentials, you must generate the encryption key, which is then embedded in the request body.

Note For more information on generating an encryption key, see Making a DELETE Request Using POSTMAN, page 6-37.

Making a DELETE Request Using POSTMAN

Step 1 Open the POSTMAN plugin in the Google Chrome browser.

Step 2 Create a new collection using the options in the left pane.

Note For more information on using the POSTMAN plugin, go to https://github.com/a85/POSTMan-Chrome-Extension/wiki.

Step 3 From the drop-down menu, choose **DELETE**.

Step 4 In the URL address bar, enter the URI.

The URI specifies the Cisco ISE server with which you are trying to communicate and the Cisco ISE resource that you are trying to access. For more information on the format of the URI, see URI, page 6-37.

Step 5 Click the **Basic Auth** tab.

The options that enable you to specify the user access credentials appear.

Step 6 Specify the access credentials in the Username and Password fields and click **Refresh Headers**.

POSTMAN displays an Authorization header with an encryption key.

Step 7 Add an accept header by specifying the following value:
application/vnd.com.cisco.ise.ers.<namespace>.<ise resource>.1.0+xml

Note For more information on the Accept Header, see Accept Header, page 6-37.

Step 8 Click **Send**.

The POSTMAN plugin displays a 200 OK status response indicating that the request is successful. You can log on to Cisco ISE to verify whether the resource you added appears in the GUI.

Java External RESTful Services Demo Application

The External RESTful Services Java Demo application is a simple External RESTful Services client code that uses the External RESTful Services API calls to configure internal users and endpoints. The main Cisco ISE dependency for this Java client code is the ers-schema.jar file. The Java Demo application consists of a generic HTTP client that is based on the Apache DefaultHttpClient and two main classes. These classes execute REST calls in a sequential order.

The source code and the required libraries are available for your reference in the demo application zip file. You can download the Java External RESTful Services Demo application and the ers-schema.jar file from the External RESTful Services Online SDK.

Related Topics

- Downloading the Java External RESTful Services Demo Application, page 6-38

Downloading the Java External RESTful Services Demo Application

Step 1 Enter the External RESTful Services SDK URL in the address bar of your browser. For example, *https://<ise hostname or ip address>:9060/ers/sdk*, where *<ise hostname or ip address>* denotes the ip address of the Cisco ISE host.

Step 2 Enter the username and case-sensitive password that was specified and configured during the initial Cisco ISE setup.

Step 3 Click **Login** or press **Enter**.

Step 4 From the Navigation pane of the External RESTful Services Online SDK page, choose **Developer Resources > Downloads**.

Step 5 Click **Schema Jar (ers-schema-1.2.0-896.jar)**.

The *ers-schema-1.0-892.jar* file is downloaded to your local machine.

Step 6 Click **Java ERS Demo Application**.

The *ers-demo-app.zip* file is downloaded to your local machine.

After you Finish

Note Refer to the *readme.txt* file that is available in the demo application zip file (ers-demo-app.zip) for all the information that is required to use the Java External RESTful Services demo application.

Using the Failure Reasons Report

You access the Cisco ISE Failure Reasons Report to view and edit a complete list of reasons why a Monitoring node operation failed. You must log in to the Cisco ISE user interface of the target Monitoring node and access the Failure Reason Editor to see the list of operation failures and possible resolutions.

Note For more information about Cisco ISE failure reasons or for general troubleshooting issues, see, "Monitoring and Troubleshooting" and "Troubleshooting Cisco ISE" in *Cisco Identity Services Engine User Guide, Release 1.2*

Viewing Failure Reasons

Step 1 Choose **Operations > Reports > Authentication Summary** report.

Step 2 In the navigation panel, expand **Monitoring** and select **Failure Reasons Editor**.

Step 3 Choose **Failure Reasons** from the list of filters provided.

Step 4 Specify the failure reason that you are looking for.

Step 5 Click **Run**.

A list of failure reasons appears in the right panel.

Step 6 Click any failure reason to get a detailed report in a new window.