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 xxxxxxxxxxxxxxxxxxx
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

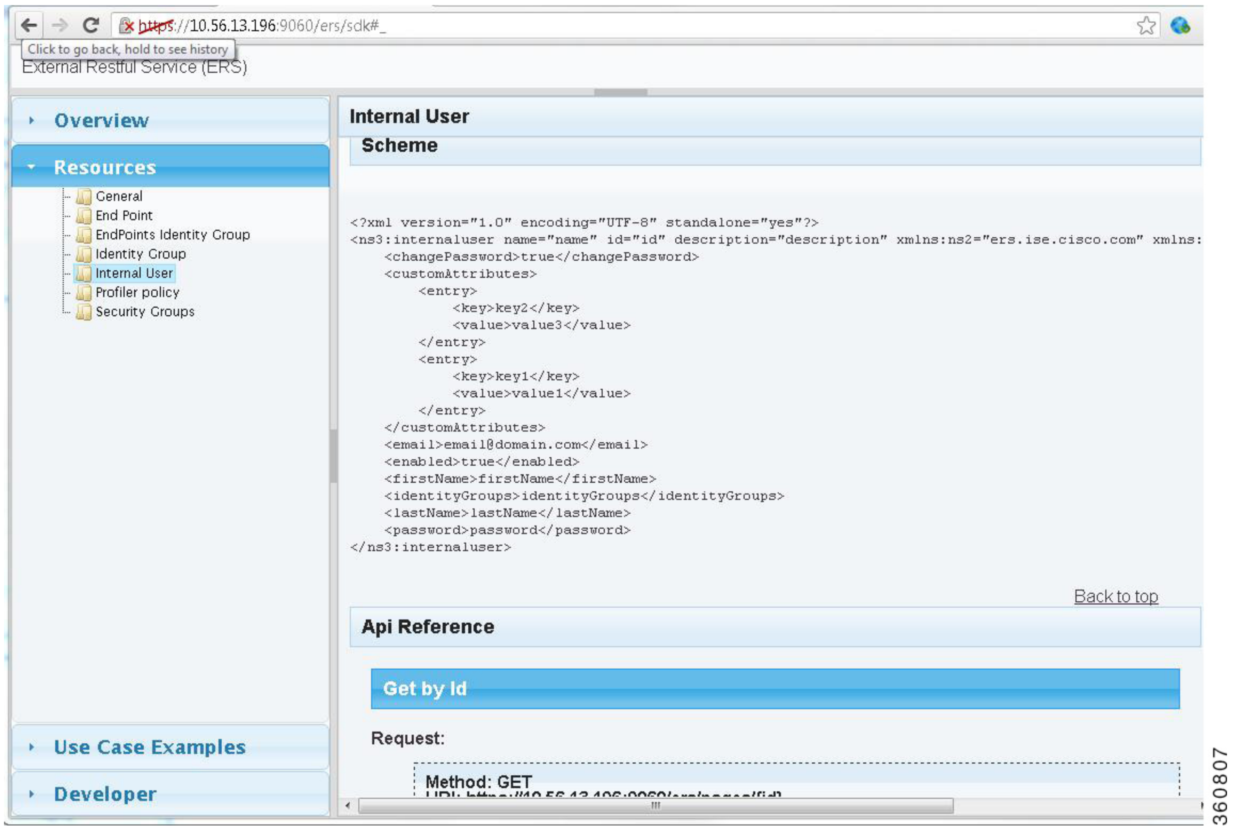
You can use the External RESTful Services software developer's kit (SDK) page to build your own tools. You access the page from the following URL: https://<ipaddress>:9060/ers/sdk

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



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

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

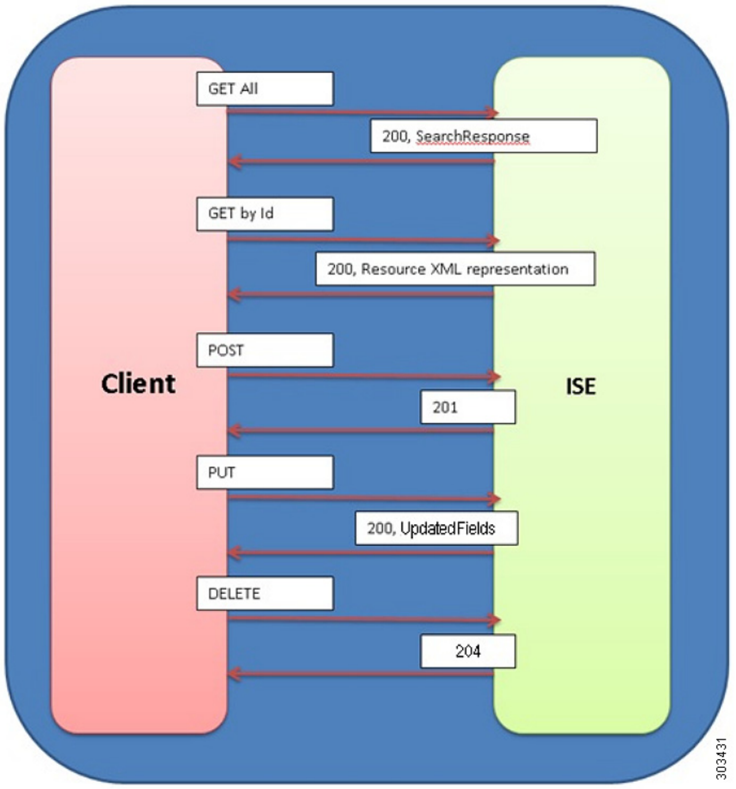
Common External RESTful Services flows always consist of an HTTPS request sent from a client and an HTTPS response from the server. The flows differ by request types, URIs, request headers, response headers, and response contents.

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

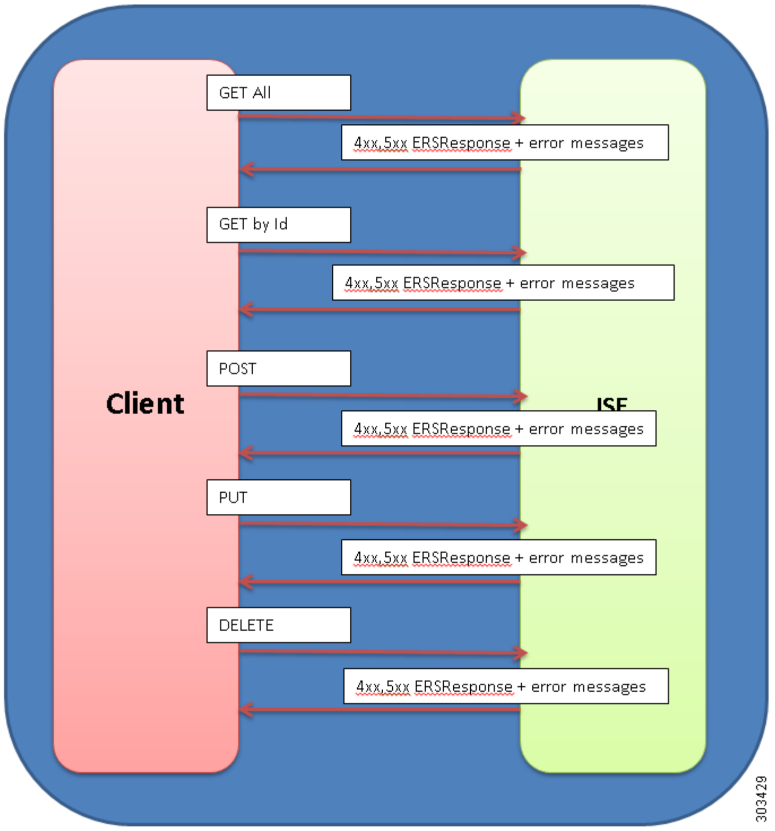
External RESTful Services Failure Flow Sequence

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


Client
ISE
GET All
4xx,5xx ERSResponse + error messages
GET by Id
4xx,5xx ERSResponse + error messages
POST
4xx,5xx ERSResponse + error messages
PUT
4xx,5xx ERSResponse + error messages
DELETE
4xx,5xx ERSResponse + error messages
303429

Related Topics

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