Introducing the Cisco IOS XE REST API

Introduction

The Representation State Transfer APIs (REST APIs) provide an alternative method to the Cisco IOS XE CLI for provisioning selected functions.

Feature History and Supported Platforms

For each Cisco IOS XE release supporting the REST API, the following table describes:

- Select new features
- Added platform support

Table 1-1 Feature History and Platform Support

Release	New Features	New Platforms Supported
3.16S	Support on Cisco ASR 1000 Series for "dual IP bring-up," using either data plane interface or management plane interface. See the software configuration guide.	ASR 1000 Series Route Processor 2 (ASR 1000-RP2)
3.14S	Support for IPv6 addressing on an interface	ASR 1001-X ASR 1002-X

Table 1-1 Feature History and Platform Support

Release	New Features	New Platforms Supported
3.13S	Additional VRF Resources VRF-Aware DNS, OSPF routing, BGP routing, EIGRP routing, Routing Table VRF-Aware NAT Saving the REST API configuration file Configuring the VPN site-to-site tunnel state Support for Locator ID Separation Protocol (LISP) Support for QoS	
3.12S	Smart License Call-Home Reload VRF support for DHCP and VPN	
3.11S	Banner BGP Best path selection Logging SNMP server TACACS server IKE keep-alive VRF support for NTP, static route, TACACs, and logging EzVPN Fall-over option for BGP neighbor API Improved configuration of user account passwords Improved configuration of interfaces: ICMP redirects, proxy ARP, unicast source verification Improved configuration of ACL Subinterface	

Table 1-1 Feature History and Platform Support

Release	New Features	New Platforms Supported
3.10S	Global configuration DNS NTP IP interfaces **Note** IPv6 for REST API is not supported in Cisco IOS XE 3.10S. DHCP Server and Relay Agent Routing Protocols: • BGP • EIGRP • OSPF ACL NAT VPN Firewall inspection IP security Site-to-Site VPN Cisco CSR 1000V software licensing Cisco CSR 1000V memory and CPU usage reports	CSR 1000V

Getting Started

You need to first configure the platform to support management using the REST API. For more information, see the configuration guide for your platform. Examples:

- Cisco CSR 1000V Series

  "Configuring Support for Management Using the REST API" section of the *Cisco CSR 1000V Series Cloud Services Router Software Configuration Guide*

- Cisco ASR 1000 Series

  "Configuring Support for Management Using the REST API" section of the *Cisco ASR 1000 Series Aggregation Services Routers Software Configuration Guide*

Important Notes

Cisco ASR1001-X and ASR1002-X Platforms—Management Port Limitation

On Cisco ASR1001-X and ASR1002-X platforms, the REST API is not supported on the management port (G0).

Known Issue with Self-Signed Certificates

There is a known issue in IOS which does not allow import and replace of an existing self-signed certificate. As a result, any running configuration being imported will fail if it contains a self-signed certificate.

Requirements for Using Firewall and VPN REST APIs

Using Firewall and VPN REST APIs requires the necessary technology package licensing for the platform.

Conventions

Cisco IOS XE REST API Request Methods

The Cisco IOS XE REST API uses the HTTP request methods described in Table 1-2.

Note All REST API requests and responses must be in JSON format. XML is not supported.

The JSON values of the *type* string should be in double-quotes. Values of type Boolean or Number should not be in double-quotes. The Boolean values are **true** or **false** in lower-case.

Table 1-2 HTTP Request Methods

HTTP Request Method	Description
GET	Retrieves the specified resource or representation. GET is a read-only operation that does not change the engine state or have any side effects. • The HTTP GET operation should not have a request body. If information is passed in a GET request, query parameters should be used instead. • Unless specified, the HTTP GET operation returns the configured state. An HTTP GET operation of the global routing table returns the dynamic run-time state.
POST	Submits data to be processed to the specified resource. The data to be processed is included in the request body. A POST operation can create a new resource. • The POST operation request contains the details of a new resource that is created in JSON. • Every POST request must include a JSON body. • For all POST operations to create a new resource, the Location header in the HTTP response contains the complete URL to be used for subsequent PUT, GET, and delete commands. • The HTTP POST response to a Create request must have a 201 return code and a Location header containing the URI of the newly created resource in the HTTP header.
PUT	Updates the specified resource with new information. The data that is included in the PUT operation replaces the previous data. • The PUT operation is used to replace or modify an existing resource. The PUT operation cannot be used to create a new resource. • The request body of a PUT operation must contain the complete representation of the mandatory attributes of the resource.
DELETE	Deletes a resource. If you delete a resource that has already been deleted, a 404 Not Found response is returned. • The HTTP DELETE operation should not have a request body. If information is passed in a GET request, query parameters should be used instead.

REST API Error Codes and Error Representation

Properties Related to Error Codes

Property	Type	Description
error-code	number	-1

Property	Type	Description
error-message	string	A brief error description or a CLI error message.
detail	string	More detailed descriptions of error message where applicable/available.

JSON Representation of Error Response

```
{
  "error-code": {number},
  "error-message": "{string}",
  "detail": "{string}"
}
```

Example 1: JSON Error Response

```
400 Bad Request

Location: http://host/api/v1/global/dns-servers
Content-Type: application/json

{
  "error-code": -1,
  "error-message": "JSON syntax error in the request",
  "detail": "Property primary is mandatory and is not present in the request."
}
```

Example 2: JSON Error Response

```
500 Internal Server Error

Location: http://host/api/v1/global/dns-servers
Content-Type: application/json

{
  "error-code": -1,
  "error-message": "Internal communication error",
  "detail": "Time-out received while communicating with the device"
}
```

Status Codes and Error Handling

The Cisco IOS XE REST API uses standard HTTP status codes to report the success or failure of the submitted requests:

- HTTP status codes from 200-299 indicate success
- HTTP status codes 400 and higher indicate failure

Table 3 describes the supported HTTP status codes and descriptions.

Table 3 HTTP Status Codes and Descriptions

Code	Status Reason	Description
200	OK	The request has succeeded.
201	Created	An asynchronous task has been completed, and the object has been created.
202	Accepted	An asynchronous task has been accepted, but the processing is not complete.
204	Accepted but with no JSON body	An HTTP GET request is successful, but the response body does not have any data
400	Bad Request	An invalid request has been submitted. Verify that the request uses the correct syntax.
401	Unauthorized	The user is not authorized to invoke the request due to invalid authentication parameters, or lack of authority.
404	Not Found	The specified resource cannot be found.
405	Method not Allowed	The HTTP verb entered is not allowed, such as a POST on a read-only resource.
500	Internal Server Error	The request failed, and no other information is available.
503	Service Unavailable	The service is not up due to internal maintenance or an outage.

Deploying REST API Using cURL: Example

The following is an example of deploying a REST API using cURL. The example shows the REST API using the POST, PUT, GET, DELETE request methods for a NAT pool.

```
[cisco@axp-4-7835-lnx ~]$ curl -v -X POST
https://172.19.153.222/api/v1/auth/token-services -H "Accept:application/json" -u
"cisco:cisco" -d "" --insecure -3
* About to connect() to 172.19.153.222 port 443
*   Trying 172.19.153.222... * connected
* Connected to 172.19.153.222 (172.19.153.222) port 443
* successfully set certificate verify locations:
*   CAfile: /usr/share/ssl/certs/ca-bundle.crt
  CApath: none
* SSL connection using AES256-SHA
* Server certificate:
*   subject: /CN=IOS-Self-Signed-Certificate-3474095688
*   start date: 2013-06-04 13:36:48 GMT
*   expire date: 2020-01-01 00:00:00 GMT
*   common name: IOS-Self-Signed-Certificate-3474095688 (does not match '172.19.153.222')
*   issuer: /CN=IOS-Self-Signed-Certificate-3474095688
* SSL certificate verify result: error number 1 (18), continuing anyway.
* Server auth using Basic with user 'cisco'
> POST /api/v1/auth/token-services HTTP/1.1
Authorization: Basic Y2lzY286Y2lzY28=
User-Agent: curl/7.12.1 (i686-redhat-linux-gnu) libcurl/7.12.1 OpenSSL/0.9.7a zlib/1.2.1.2
libidn/0.5.6
Host: 172.19.153.222
```

```
Pragma: no-cache
Accept:application/json
Content-Length: 0
Content-Type: application/x-www-form-urlencoded

< HTTP/1.1 201 Created
< Content-Type: application/json
< Content-Length: 204
< Date: Thu, 06 Jun 2013 09:05:31 GMT
< Server: cisco-IOSd..
* Connection #0 to host 172.19.153.222 left intact
* Closing connection #0
{"kind": "object#auth-token", "expiry-time": "Thu Jun  6 02:20:29 2013", "token-id":
"9qAm/T0etz5Bj84H2j+nkxC7aGmQ9rNxsgYsaQho5u8=", "link":
"https://172.19.153.222/api/v1/auth/token-services/2257880484"}[cisco@axp-4-7835-lnx ~]$
[cisco@axp-4-7835-lnx ~]$
[cisco@axp-4-7835-lnx ~]$ curl -v -H "Accept:application/json" -H "X-Auth-Token:
9qAm/T0etz5Bj84H2j+nkxC7aGmQ9rNxsgYsaQho5u8=" -H "content-type: application/json" -X POST
https://172.19.153.222/api/v1/nat-svc/pool -d '{"nat-pool-id": "test4-nat-pool",
"start-ip-address": "172.16.10.1", "end-ip-address": "172.16.10.63", "prefix-length": 32}'
--insecure -3
* About to connect() to 172.19.153.222 port 443
*   Trying 172.19.153.222... * connected
* Connected to 172.19.153.222 (172.19.153.222) port 443
* successfully set certificate verify locations:
*   CAfile: /usr/share/ssl/certs/ca-bundle.crt
  CApath: none
* SSL connection using AES256-SHA
* Server certificate:
*   subject: /CN=IOS-Self-Signed-Certificate-3474095688
*   start date: 2013-06-04 13:36:48 GMT
*   expire date: 2020-01-01 00:00:00 GMT
*   common name: IOS-Self-Signed-Certificate-3474095688 (does not match '172.19.153.222')
*   issuer: /CN=IOS-Self-Signed-Certificate-3474095688
* SSL certificate verify result: error number 1 (18), continuing anyway.
> POST /api/v1/nat-svc/pool HTTP/1.1
User-Agent: curl/7.12.1 (i686-redhat-linux-gnu) libcurl/7.12.1 OpenSSL/0.9.7a zlib/1.2.1.2
libidn/0.5.6
Host: 172.19.153.222
Pragma: no-cache
Accept:application/json
X-Auth-Token: 9qAm/T0etz5Bj84H2j+nkxC7aGmQ9rNxsgYsaQho5u8=
content-type: application/json
Content-Length: 123

{"nat-pool-id": "test4-nat-pool", "start-ip-address": "172.16.10.1", "end-ip-address":
"172.16.10.63", "prefix-length": 32}< HTTP/1.1 201 Created
< Content-Type: application/json
< Content-Length: 4
< Location: https://172.19.153.222/api/v1/nat-svc/pool/test4-nat-pool
< Date: Thu, 06 Jun 2013 09:09:27 GMT
< Server: cisco-IOSd..
* Connection #0 to host 172.19.153.222 left intact
* Closing connection #0
null[cisco@axp-4-7835-lnx ~]$
[cisco@axp-4-7835-lnx ~]$
[cisco@axp-4-7835-lnx ~]$ curl -v -H "Accept:application/json" -H "X-Auth-Token:
9qAm/T0etz5Bj84H2j+nkxC7aGmQ9rNxsgYsaQho5u8=" -H "content-type: application/json" -X PUT
https://172.19.153.222/api/v1/nat-svc/pool/test4-nat-pool -d '{"nat-pool-id":
"marketing-nat-pool", "start-ip-address": "1.16.10.17", "end-ip-address": "1.16.10.57",
"prefix-length": 16}' --insecure -3
* About to connect() to 172.19.153.222 port 443
*   Trying 172.19.153.222... * connected
* Connected to 172.19.153.222 (172.19.153.222) port 443
```

```
* successfully set certificate verify locations:
*   CAfile: /usr/share/ssl/certs/ca-bundle.crt
  CApath: none
* SSL connection using AES256-SHA
* Server certificate:
*   subject: /CN=IOS-Self-Signed-Certificate-3474095688
*   start date: 2013-06-04 13:36:48 GMT
*   expire date: 2020-01-01 00:00:00 GMT
*   common name: IOS-Self-Signed-Certificate-3474095688 (does not match '172.19.153.222')
*   issuer: /CN=IOS-Self-Signed-Certificate-3474095688
* SSL certificate verify result: error number 1 (18), continuing anyway.
> PUT /api/v1/nat-svc/pool/test4-nat-pool HTTP/1.1
User-Agent: curl/7.12.1 (i686-redhat-linux-gnu) libcurl/7.12.1 OpenSSL/0.9.7a zlib/1.2.1.2
libidn/0.5.6
Host: 172.19.153.222
Pragma: no-cache
Accept:application/json
X-Auth-Token: 9qAm/T0etz5Bj84H2j+nkxC7aGmQ9rNxsgYsaQho5u8=
content-type: application/json
Content-Length: 124

{"nat-pool-id": "marketing-nat-pool", "start-ip-address": "1.16.10.17", "end-ip-address":
"1.16.10.57", "prefix-length": 16}
< HTTP/1.1 204 No Content
< Content-Type: application/json
< Date: Thu, 06 Jun 2013 09:13:19 GMT
< Server: cisco-IOSd..
* Connection #0 to host 172.19.153.222 left intact
* Closing connection #0
[cisco@axp-4-7835-lnx ~]$
[cisco@axp-4-7835-lnx ~]$ curl -v -H "Accept:application/json" -H "X-Auth-Token:
9qAm/T0etz5Bj84H2j+nkxC7aGmQ9rNxsgYsaQho5u8=" -H "content-type: application/json" -X GET
https://172.19.153.222/api/v1/nat-svc/pool/test4-nat-pool --insecure -3
* About to connect() to 172.19.153.222 port 443
*   Trying 172.19.153.222... * connected
* Connected to 172.19.153.222 (172.19.153.222) port 443
* successfully set certificate verify locations:
*   CAfile: /usr/share/ssl/certs/ca-bundle.crt
  CApath: none
* SSL connection using AES256-SHA
* Server certificate:
*   subject: /CN=IOS-Self-Signed-Certificate-3474095688
*   start date: 2013-06-04 13:36:48 GMT
*   expire date: 2020-01-01 00:00:00 GMT
*   common name: IOS-Self-Signed-Certificate-3474095688 (does not match '172.19.153.222')
*   issuer: /CN=IOS-Self-Signed-Certificate-3474095688
* SSL certificate verify result: error number 1 (18), continuing anyway.
> GET /api/v1/nat-svc/pool/test4-nat-pool HTTP/1.1
User-Agent: curl/7.12.1 (i686-redhat-linux-gnu) libcurl/7.12.1 OpenSSL/0.9.7a zlib/1.2.1.2
libidn/0.5.6
Host: 172.19.153.222
Pragma: no-cache
Accept:application/json
X-Auth-Token: 9qAm/T0etz5Bj84H2j+nkxC7aGmQ9rNxsgYsaQho5u8=
content-type: application/json

< HTTP/1.1 200 OK
< Content-Type: application/json
< Content-Length: 147
< Date: Thu, 06 Jun 2013 09:13:24 GMT
< Server: cisco-IOSd..
* Connection #0 to host 172.19.153.222 left intact
* Closing connection #0
```

```
{"nat-pool-id": "test4-nat-pool", "kind": "object#nat-pool", "prefix-length": 16,
"end-ip-address": "1.16.10.57", "start-ip-address": "1.16.10.17"}[cisco@axp-4-7835-lnx
~]$
[cisco@axp-4-7835-lnx ~]$
[cisco@axp-4-7835-lnx ~]$ curl -v -H "Accept:application/json" -H "X-Auth-Token:
9qAm/T0etz5Bj84H2j+nkxC7aGmQ9rNxsgYsaQho5u8=" -H "content-type: application/json" -X
DELETE https://172.19.153.222/api/v1/nat-svc/pool/test4-nat-pool --insecure -3
* About to connect() to 172.19.153.222 port 443
*   Trying 172.19.153.222... * connected
* Connected to 172.19.153.222 (172.19.153.222) port 443
* successfully set certificate verify locations:
*   CAfile: /usr/share/ssl/certs/ca-bundle.crt
  CApath: none
* SSL connection using AES256-SHA
* Server certificate:
*   subject: /CN=IOS-Self-Signed-Certificate-3474095688
*   start date: 2013-06-04 13:36:48 GMT
*   expire date: 2020-01-01 00:00:00 GMT
*   common name: IOS-Self-Signed-Certificate-3474095688 (does not match '172.19.153.222')
*   issuer: /CN=IOS-Self-Signed-Certificate-3474095688
* SSL certificate verify result: error number 1 (18), continuing anyway.
> DELETE /api/v1/nat-svc/pool/test4-nat-pool HTTP/1.1
User-Agent: curl/7.12.1 (i686-redhat-linux-gnu) libcurl/7.12.1 OpenSSL/0.9.7a zlib/1.2.1.2
libidn/0.5.6
Host: 172.19.153.222
Pragma: no-cache
Accept:application/json
X-Auth-Token: 9qAm/T0etz5Bj84H2j+nkxC7aGmQ9rNxsgYsaQho5u8=
content-type: application/json

< HTTP/1.1 204 No Content
< Content-Type: application/json
< Date: Thu, 06 Jun 2013 09:13:50 GMT
< Server: cisco-IOSd..
* Connection #0 to host 172.19.153.222 left intact
* Closing connection #0
[cisco@axp-4-7835-lnx ~]$
```