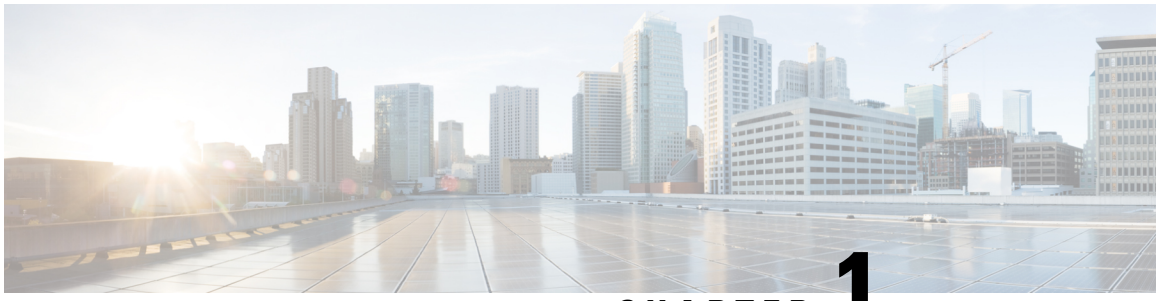
Cisco Expressway REST API Summary Guide (X14.2)

First Published: 2022-08-10

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



CHAPTER 1

Using the Expressway REST API

Change History

Table 1: Change History

Date	Change	Reason
August 2022	First published for X14.2. Introduced APIs to upgrade a single node.	X14.2 release
April 2020	Status API support added along with provisioning APIs. Changed the base URL to access the Expressway REST API to http://<external_address>/api.	Document correction
January 2019	Removed software version from document, as it is no longer version-specific.	Documentation correction
September 2018	Updated software version from X8.11 to X8.11.1, as version X8.11 is no longer available.	Software withdrawn
July 2018	Removed details about individual API calls, as the API is self-documented.	X8.11 release
July 2017	Phase three of REST API. Now includes firewall rules, SIP, and domain certificates.	X8.10 release
January 2017	Updated with HTTP allow list calls and get by filter option.	X8.9.1 release

Date	Change	Reason
December 2016	Phase two of REST API. Now includes B2B functionality and ability to delete.	X8.9 release
June 2016	First phase of REST API to set up Mobile and Remote Access (MRA).	X8.8 release

Using the Expressway REST API

The Expressway REST API is compliant with RAML version 0.8 (raml.org/spec.html). Although the API is fully compliant, it does not support nested APIs.

The API is self-documented using RESTful API Modeling Language (RAML). You can access the RAML definitions for your system at https://<Expressway FQDN or IP address>/api/raml. An experimental schema browser is embedded in the web user interface, and can be accessed from the **Experimental** menu.

Schemas

All request and response schema on the Expressway REST API use JSON Schema version 4 (json-schema.org/documentation.html). Request parameters are not supported and only JSON schemas are used.

Authentication

The API is only accessible via HTTPS and requires authentication. The authentication credentials are the administrator credentials on the Expressway node.

Base URL

The base URL to access the Expressway REST API is http://<external_address>/api. For example, to access the system information, use https://10.0.0.1/api/provisioning/sysinfo.

The REST API is published in the following categories:

- Cisco Expressway-E

  /provisioning/edge/ <remaining path> (for example, https://10.0.0.1/api/provisioning/edge/zone/traversalserver)

- Cisco Expressway-C

  /provisioning/controller/ <remaining path> (for example, https://10.0.0.1/api/provisioning/controller/zone/traversalclient)

- Common between Cisco Expressway-E and Cisco Expressway-C

  /provisioning/common/<remaining path> (for example, https://10.0.0.1/api/provisioning/common/adminaccount/changepassword)

  Some maintenance-related items like restart and system information are standalone calls and do not apply to any of the categories.

You can also filter Get requests to find a specific entry. For example, /controller/zone/traversalclient/name/myzone returns the traversal client zone called "myzone".

- REST APIs which get the status of functionalities as status/common/<remaining path> (for example, http://10.0.0.1/api/status/common/smartlicensing/licensing

Sample requests and responses

This section provides examples on how to use Expressway API methods. The examples relate to API methods for the DNS server and NTP server.

Example: USING APIs to UPGRADE EXPRESSWAY

Add SFTP information

This example adds SFTP information using JSON API

URL	PUT https://<Expressway FQDN or IP address>/api/v1/provisioning/common/sftpconfig
Request body	{ "SftpServer": "10.0.0.0", "Username": "username", "Password": "password", "RsaKey": "*****" }
Response body	{ "Message": "Update successful" }

For “RsaKey” value, recommended way is to obtain RSA public key from the SFTP Server Admin. Alternatively, use the below command from a system in the network (that can reach your SFTP Server) to fetch the public key.

```
ssh-keyscan <IP of SFTP server> | grep "ssh-rsa" --color
```

This example adds SFTP configuration information using cURL

```
curl -X PUT -k -i 'https:// <Expressway FQDN or IP
address>/api/v1/provisioning/common/sftpconfig' --data '{SftpServer": "10.0.0.0", “Username”:
 "username", “Password”: "password", “RsaKey”: "*****"}'
```

GET request is also supported.

```
curl -X GET -k -i 'https:// <Expressway FQDN or
IPaddress>/api/v1/provisioning/common/sftpconfig'
```

Trigger upgrade

This example triggers upgrade using JSON API

URL	POST https://<Expressway FQDN or IP address>/api/v1/provisioning/common/upgrade
Request body	{ "ImageDownloadMode": "sftp", "UpgradeFileLocationSftpPath": "/home/username/upgrade", "UpgradeFileName": "oak.tgz",

	<pre>"UpgradeFileSha512": "*****",
	"ClusterUpgrade": "no",
	"WaitForCallDisconnect": 0,
	"AutoReboot": "yes"
	}</pre>

Response body	<pre>{
	"Message": "Upgrade request created successfully"
	}</pre>

Note Currently, “ImageDownloadMode” only supports the value “sftp”, and “ClusterUpgrade” only supports “no”.

This example triggers upgrade using cURL.

```
curl -X POST -k -i 'https:// <Expressway FQDN or IP
address>/api/v1/provisioning/common/upgrade' --data '{"ImageDownloadMode": "sftp",
"UpgradeFileLocationSftpPath": "/home/username/upgrade", "UpgradeFileName": "oak.tgz",
"UpgradeFileSha512": "*****", "ClusterUpgrade": "no", "WaitForCallDisconnect": 0,
"AutoReboot": "yes"}’
```

Get upgrade status information

This example retrieves upgrade status information using JSON API

URL	GET https://<Expressway FQDN or IP address>/api/v1/status/common/upgradestatus
Request body	This operation does not require a request body.

Response body	<pre>[
	{
	"StatusUpdateTime": "22-05-18T13-23-36",
	"UpgradeStatus": "UPGD_STARTED"
	}
	]</pre>

This example retrieves upgrade status information using cURL.

```
curl -X GET -k -i 'https:// <Expressway FQDN or IPaddress>/api/v1/status/common/upgradestatus'
```

‘**UpgradeStatus**’ responses:

- UPGD_INITIATED (All SFTP details - sftp server (IP or FQDN), username, password and rsakey should be present)
- UPGD_FAILED (If SFTP server is not configured)
- UPGD_FILE_DOWNLOADING
- UPGD_FILE_DOWNLOADED (Time taken depends on file speed transfer between the system and SFTP server)
- UPGD_FILE_DOWNLOADFAILED

  Failure reasons

  - If SFTP configuration is incorrect

- If SFTP server connection times out (30 seconds)
- If SHA512sum check failed

- UPGD_WAIT_FOR_CALLDISCONNECT (Wait time based on configured value in upgrade API)
- UPGD_STARTED
- UPGD_REBOOT (Timeout is 15 mins after UPGD_STARTED)
- UPGD_FAILED

  Failed reasons:

  - Install framework failure
  - Install framework failure - no upgrade flags created

- UPGD_SUCCESSFUL (After reboot)

Example: USING API FOR DNS SERVER

Retrieve DNS Server information

This example retrieves the DNS server information using JSON API.

URL	GET https://<Expressway FQDN or IP address>/api/v1/provisioning/common/dns/dnsserver
Request body	This operation does not require a request body.
Response body	{ "DefaultDNSServers": { "index": 2, "address": "10.0.0.2" } }

This example retrieves the DNS server information using cURL.

```
curl -X GET -k -i 'https://<Expressway FQDN or IP
address>/api/v1/provisioning/common/dns/dnsserver'
```

Add DNS server

This example adds a DNS server with an IP address 10.0.0.2 and index value 2 using JSON API.

URL	POST https://<Expressway FQDN or IP address>/api/v1/provisioning/common/dns/dnsserver'
Request body	{ "DefaultDNSServers": { "index": 2, "address": "10.0.0.2" } }

Response body	{
	"Message": "The operation was successful"
	}

This example adds a DNS server with an IP address 10.0.0.2 and index value 2 using cURL.

```
curl -X POST -k -i 'https://<Expressway FQDN or IP
address>/api/v1/provisioning/common/dns/dnsserver' --data '{"DefaultDNSServers": {"index":
 2, "address": "10.0.0.2"}}'
```

Modify DNS server

This example modifies the IP address of the DNS server with the index value 2 using JSON API.

URL	PUT https://<Expressway FQDN or IP address>/api/v1/provisioning/common/dns/dnsserver

Request body	{
	"DefaultDNSServers":
	{
	"index": 2,
	"address": "10.0.0.3"
	}
	}

Response body	{
	"Message": "The operation was successful"
	}

This example modifies the IP address of the DNS server with the index value 2 using cURL.

```
curl -X PUT -k -i 'https://<Expressway FQDN or IP
address>/api/v1/provisioning/v1/common/dns/dnsserver' --data '{"DefaultDNSServers": {"index":
 2, "address": "10.0.0.3"}}'
```

Delete DNS server

This example deletes the DNS server with the index value of 2 using JSON API.

URL	DELETE https://<Expressway FQDN or IP address>/api/v1/provisioning/common/dns/dnsserver

Request body	{
	"index": 2
	}

Response body	{
	"Message": "The operation was successful"
	}

This example deletes the DNS server with the index value of 2 using cURL.

```
curl -X DELETE -k -i 'https://<Expressway FQDN or IP
address>/api/v1/provisioning/common/dns/dnsserver' --data '{"index": 2}'}'
```

Example: USING API FOR NTP SERVER

Retrieve NTP Server information

This example retrieves the NTP server information using JSON API.

URL	GET https://<Expressway FQDN or IP address>/api/v1/provisioning/common/time/ntpserver
Request body	This operation does not require a request body.

Response body	<pre>{
	"index": 5,
	"KeyId": 1,
	"Hash": "sha1",
	"Authentication": "disabled",
	"Address": "10.0.0.1"
	}</pre>

This example retrieves the NTP server information using cURL.

```
curl -X GET -k -i '<Expressway FQDN or IP address>/api/v1/provisioning/common/time/ntpserver'
```

Add NTP Server

This example adds an NTP server with an IP address 10.0.0.2 using JSON API.

URL	POST https://<Expressway FQDN or IP Address>/api/v1/provisioning/common/time/ntpserver

Request body	<pre>{
	"index": 6,
	"Address": "10.0.0.2",
	"KeyId": 1,
	"Hash": "sha1",
	"Authentication": "disabled"
	}</pre>

Response body	<pre>{
	"Message": "The operation was successful"
	}</pre>

This example adds an NTP server with an IP address 10.0.0.2 using cURL.

```
curl -X POST -k -i 'https://<Expressway FQDN or IP
address>/api/v1/provisioning/common/time/ntpserver' --data '{"index": 6, "Address":
"10.0.0.2", "KeyId": 1, "Hash": "sha1", "Authentication": "disabled"}'
```

Modify NTP Server information

This example modifies the IP address of the NTP server with the index value 6 using JSON API.

URL	PUT https://<Expressway FQDN or IP address>/api/v1/provisioning/common/time/ntpserver

Request body	<pre>{
	"index": 6,
	"Address": "10.0.0.3",
	"KeyId": 1,
	"Hash": "sha1",
	"Authentication": "disabled"
	}</pre>

Response body	<pre>{
	"Message": "The operation was successful"
	}</pre>

This example modifies IP address of the NTP server with the index value 6 using cURL.

```
curl -X POST -k -i 'https://<Expressway FQDN or IP
address>/api/v1/provisioning/common/time/ntpserver' --data '{"index": 6, "Address":
"10.0.0.3", "KeyId": 1, "Hash": "sha1", "Authentication": "disabled"}'
```

Delete NTP Server

This example deletes the NTP server with the index value of 6 using JSON API.

URL	DELETE https://<Expressway FQDN or IP address>/api/v1/provisioning/common/time/ntpserver
Request body	`{ "index": 6 }`
Response body	`{ "Message": "The operation was successful" }`

This example deletes the DNS server with the index value of 6 using cURL.

```
curl -X DELETE -k -i 'https://<Expressway FQDN or IP
address>/api/v1/provisioning/common/time/ntpserver' --data '{"index": 6}'}'
```

Example: USING API FOR RETRIEVING SMART LICENSING STATUS

Retrieve Smart Licensing Status

This example retrieves Smart Licensing Status.

URL	GET https://<Expressway FQDN or IP address>/api/v1/status/common/smartlicensing/licensing
Request body	This operation does not require a request body.
Response body	`{ "ExportControlledFunctionality": "True", "VirtualAccount": "Expressway", "SmartAccount": "testaccount.cisco.com", "Authorization": { "LicenseAuthorizationStatus": "OUT OF COMPLIANCE", "AuthorizationExpires": "May, 05 May 2020 06:13:06 GMT", "NextAuthorizationAttempt": "February, 05 Feb 2020 18:18:07 GMT", "LastAuthorizationAttempt": "February, 05 Feb 2020 06:18:07 GMT" }, "Registration": { "RegistrationStatus": "REGISTERED", "InitialRegistration": "February, 05 Feb 2020 05:59:04 GMT", "RegistrationExpires": "February, 04 Feb 2021 05:54:03 GMT", "NextRenewalAttempt": "August, 03 Aug 2020 05:59:04 GMT" } }`

This example retrieves the Smart Licensing Status information using cURL.

```
curl -X GET -k -i 'https://<Expressway FQDN or IP
address>/api/v1/status/common/smartlicensing/licensing'
```

Using the Expressway REST API