CC-SG RESTful-API Programming Guide

April 2026
API Version 1.0.0

FCC Information

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a commercial installation. This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. Operation of this equipment in a residential environment may cause harmful interference.

VCCI Information (Japan)

この装置は，クラスA情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。 VCCI－A

Raritan is not responsible for damage to this product resulting from accident, disaster, misuse, abuse, non-Raritan modification of the product, or other events outside of Raritan's reasonable control or not arising under normal operating conditions.

If a power cable is included with this product, it must be used exclusively for this product.

CE c UL us LISTED 1F61 I.T.E.

Contents

Introduction

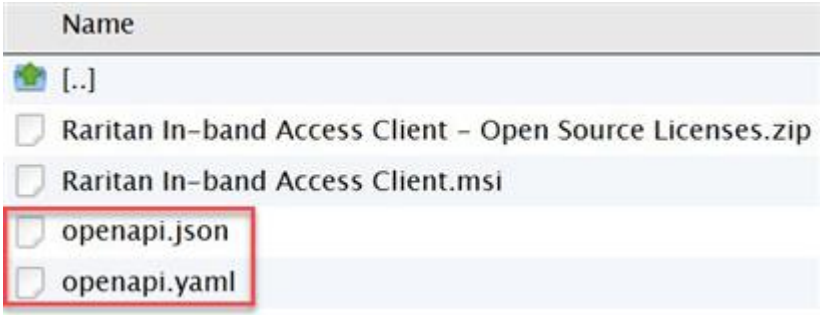
The CC-SG REST API provides all the capabilities of the WS-AP and additional functionality while aligning with modern industry standards. It enables programmatic access to CC-SG operations over HTTPS and does not require any special licensing or additional configuration on the CC-SG to use.

Getting Started

You can download the OpenAPI specification for Restful API from your CC-SG at: https://<CC-SG IP address>/download. It's available in two formats:

- YAML (openapi.yaml)
- JSON (openapi.json)

These files include both the machine-readable API definition and easy-to-understand English documentation. You can use any development tools you prefer to view the specification or to generate a client library for your application.



When you're ready to start using the API, make sure your client is configured with this base URL: https://CC-SG IP address/CommandCenterWeb/

Authentication

A session ID is needed to send any API call.

► *To get the session ID:*

Use a POST request with your username and password to: https://CC-SG IP address/CommandCenterWeb/rest/v1/session

► *To close the session :*

Use DELETE request to the same URL session.

Example-(Python request):

This example uses openapi-python-client, a Python module that automatically generates Python code to interact with an API, and demonstrates how to authenticate, retrieve a list of active sessions, and sign out.

```
from cc_sg_rest_api_client import Client
from cc_sg_rest_api_client.api.default import (
delete_rest_v1_session,
post_rest_v1_session,
get_rest_v1_session_sessions
)
from cc_sg_rest_api_client.models import Session, Credential
address = "CCSGADDRESS"
# For testing, accept self-signed server cert

verify_ssl = False
# Create client with base URL

client = Client(
   base_url=f"https://{address}/CommandCenterWeb/",
   verify_ssl=verify_ssl
   )
# Sign on
cred = Credential()
cred.user = "user"
cred.password = "password"
response = post_rest_v1_session.sync_detailed(
   client=client,
   body=cred
   )
print(f"Response: {response.status_code}")
# Retrieve active sessions

response = get_rest_v1_session_sessions.sync_detailed(
   client=client
   )
print(f"Response: {response.status_code}")
print("Sessions result:")
for s in response.parsed:
if s.is_caller:
```

```
print("\tCaller's session")
print(
f"\t{s.id}\n"
f"\t{s.address}",
f"{s.time},",
f"{s.type_},
f"{s.user},
f"{s.user_id}\n"
)
# Sign out
response = delete_rest_v1_session.sync_detailed(
   client=client
   )
print(f"Response: {response.status_code}")
```

Note: The CC-SG returns the session identifier in a cookie. When you access the API through a web browser, the session ID isn't visible to you, but the browser automatically includes it in all subsequent requests on your behalf.

Migration from the WS-API

If you previously used the WS-API, the following table maps each WS operation to its corresponding REST API endpoint to help you update your client.

► *Difference between the two APIs:*

- The WS-API is largely name based.
- The REST API is primarily ID based (numeric IDs, except for sessions).

While the WS-API relies on names, the new REST API is built around identifiers (numeric IDs, except for session IDs). You can still perform name based lookups or retrieve all items using GET requests, but all POST, PUT, and DELETE operations require an ID.

To get the ID of an existing item, simply run the appropriate GET request and locate the identifier in the response. When you create a new item with a POST request, the API returns the ID of the resource it just created.

Migration from the WS-API:

WS-API	HTTP Method	URL Path	Notes
Authentication and Authorization Services			
getSessions	GET	/rest/v1/session/sessions	
signOn	POST	/rest/v1/session	

WS-API	HTTP Method	URL Path	Notes
signoff	DELETE	/rest/v1/session	
forceSignOff	DELETE	/rest/v1/session/sessions	
closeInterfaceConnections	DELETE	/rest/v1/session/connections	
getInterfaceConnections	GET	/rest/v1/session/connections	
Category Management Services			
getCategory	GET	/rest/v1/association/category	Use name parameter.
addCategory	POST	/rest/v1/association/category	
editCategory	PUT	/rest/v1/association/category/{id}	
deleteCategory	DELETE	/rest/v1/association/category/{id}	
addElementToCategory	POST	/rest/v1/association/category/{id}/element	
renameElementInCategory	PUT	/rest/v1/association/category/element/{id}	
deleteElementFromCategory	DELETE	/rest/v1/association/category/element/{id}	
Device Management Services			
getDevice	GET	/rest/v1/device	Use name parameter.
deleteDevice	DELETE	/rest/v1/device/{id}	
getDeviceStatus	GET	/rest/v1/device	Set status parameter to true.
setUserPassword	PUT	/rest/v1/device/{id}/password	
Logging Management Services			

WS-API	HTTP Method	URL Path	Notes
runReport	GET	/rest/v1/report/log/audit /rest/v1/report/log/access	Modeled after Admin client reports.
	GET	/rest/v1/report/log	Equivalent to runReport() with an empty or null messageType.
getReportRecords			
getReportDocument	GET	/rest/v1/report/device/port	
deleteReport			
Node Management Services			
getCCSGAppletURL			
getCCSGHTMLClientURL			
getInterfaceURL	GET	/rest/v1/node/interface/{id}/url	
getAccessMethodsForNode			
getAccessMethodsForNodeByInterfaceID			
getNodeByName	GET	/rest/v1/node	
getNodeByAttribute	GET	/rest/v1/node/interface/cim/{serial}	Find interface by CIM serial.
getNodeByInterfaceName	GET	/rest/v1/node	Use interface parameter.
getNodeByAssociation	GET	/rest/v1/node	Use element parameter set to the element ID.
getNodesForUser	GET	/rest/v1/node	Set name parameter to "*".

WS-API	HTTP Method	URL Path	Notes
getNodeStatus	GET	/rest/v1/node	Set status parameter to true.
addAssociationToNode	PUT	/rest/v1/node/{id}/associations	Set of elements to associate with the node.
deleteAssociationFromNode			Unneeded because PUT overwrites.
renameNode	PUT	/rest/v1/node/{id}	
getNodePower	GET	/rest/v1/node/{id}	Set power parameter to true.
setNodePower	PUT	/rest/v1/node/{id}/power/{operation} /rest/v1/node/interface/{id}/power/{operation}	Done per node or per interface.
System Management Services			
getSystemInfo	GET	/rest/v1/report/about /rest/v1/report/cluster /rest/v1/report/license	
importCSV	POST	/rest/v1/task/import/{type}	
getImportStatus	GET	/rest/v1/task/{id}/status	
deleteTask	DELETE	/rest/v1/task/{id}	Undocumented in the WS-API guide.
User Management Services			
getUser	GET	/rest/v1/user	Use name parameter.
getAllUsers	GET	/rest/v1/user	Do not pass name parameter.
addUser	POST	/rest/v1/user	
editUser	POST	/rest/v1/user/{id}	
deleteUser	DELETE	/rest/v1/user/{id}	

WS-API	HTTP Method	URL Path	Notes
addUserToGroup	PUT	/rest/v1/user/{user}/group/{group}	One group at a time.
deleteUserFromGroup	DELETE	/rest/v1/user/{user}/group/{group}	One group at a time.