Brocade® SANnav™ Management Portal REST API and Northbound Streaming Reference Manual, 2.3.x

Reference Manual
April 28, 2023

Table of Contents

Introduction to the SANnav REST API

This manual describes the REST API (a programmable web-service interface) for Brocade® SANnav™ Management Portal. The SANnav REST API can access the SANnav Management Portal server system. This manual contains definitions of REST APIs that you can use to access SANnav Management Portal, including streaming performance and flow metrics to an external server.

Within this document, SANnav Management Portal might also be referred to simply as "SANnav".

Contacting Technical Support for Your Brocade® Product

If you purchased Brocade® product support from a Broadcom® OEM or solution provider, contact your OEM or solution provider for all your product support needs.

- OEM and solution providers are trained and certified by Broadcom to support Brocade products.
- Broadcom provides backline support for issues that cannot be resolved by the OEM or solution provider.
- Brocade Supplemental Support augments your existing OEM support contract, providing direct access to Brocade expertise. For more information on this option, contact Broadcom or your OEM.
- For questions regarding service levels and response times, contact your OEM or solution provider.

If you purchased Brocade product support directly from Broadcom, use one of the following methods to contact the Technical Assistance Center 24x7. For product support information and the latest information on contacting the Technical Assistance Center, go to www.broadcom.com/support/fibre-channel-networking/contact-brocade-support.

Online	Telephone
For nonurgent issues, the preferred method is to log on to the Support portal at support.broadcom.com. (You must initially register to gain access to the Support portal.) Once registered, log on and then select **Brocade Products**. You can now navigate to the following sites: • **Case Management** • **Software Downloads** • **Licensing** • **SAN Reports** • **Brocade Support Link** • **Training & Education**	For Severity 1 (critical) issues, call Brocade Fibre Channel Networking Global Support at one of the phone numbers listed at www.broadcom.com/support/fibre-channel-networking/contact-brocade-support.

Document Feedback

Quality is our first concern. We have made every effort to ensure the accuracy and completeness of this document. However, if you find an error or an omission or if you think that a topic needs further development, we want to hear from you. Send your feedback to documentation.pdl@broadcom.com. Provide the publication title; topic heading; publication number and page number (for PDF documents); URL (for HTML documents); and as much detail as possible.

SANnav REST API Overview

SANnav Management Portal supports an application programming interface (API) for managing Brocade storage area network (SAN) fabrics. The REST API provides you with a web-services interface for accessing the SANnav Management Portal server system. The REST APIs are organized into various services, such as Login, Discovery, FCR, Fault, Inventory, Northbound Streaming, and Health Summary. You can also use third-party REST API clients to interact with SANnav Management Portal.

NOTE

SANnav Global View does not expose any external REST APIs.

SANnav REST API supports the HTTPS protocol. The default HTTPS port number is 443.

SANnav REST API can be accessed using only the IPv4 address of the host (an IPv6 address is not supported).

The SANnav REST API supports the following session types:

- SANnav REST API session-based operation
- SANnav REST API session-less operation

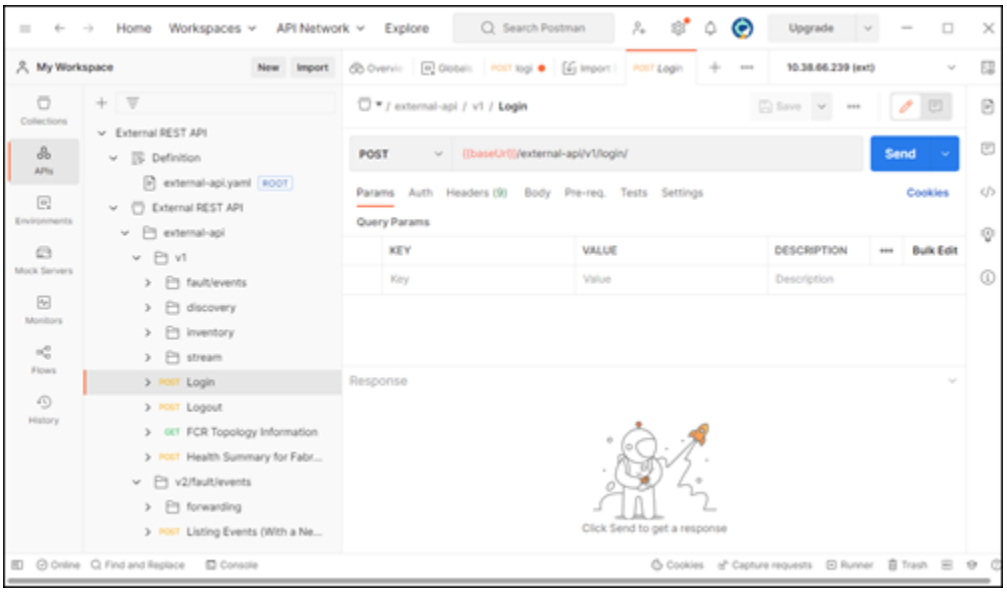
YAML Overview

The SANnav REST API services are configured using YAML 1.2 (yet another markup language), a data serialization language.

The YAML file defines the resources and shared data types. Each resource is exposed through a universal resource indicator (URI) (see Resources) and may use the shared data types (see Definitions).

The SANnav REST API YAML file (`external-api.yaml`) is in the following location: `<install_home>/conf/northbound/api` . The YAML file can be viewed by importing the file in to a REST API YAML editor.

Figure 1: SANnav External API File in a REST API Editor



Overall Strategy for the SANnav Management Portal REST API

The SANnav REST API provides functionality that is not available in the Fabric OS® REST APIs. The SANnav REST API includes support for the following SANnav features:

- List the fabrics managed by the SANnav server.
- List switches (members) in the fabric.
- Display FCR topology information for routing topologies, such as edge-to-edge, backbone-to-edge, and edge-to-backbone.
- Configure event forwarding.
- Acknowledge or unacknowledge events.
- Display a list of filtered events.
- Search the inventory.
- Display inventory data for fabrics, switches, switch ports, devices, device ports, and chassis.
- Map a device port to an enclosure.
- Stream FC and flow data (via northbound streaming).
- Display a health summary for a fabric, switch, host, or storage.

Using SANnav REST API Session-Based Operation

SANnav REST API session-based operation requires that you log on to obtain a session ID, and then provide that session ID in the REST API operation. A session ID is valid until the REST API session expires.

Keep the following in mind when using the SANnav REST API.

Before You Begin

Before you can use the SANnav REST API, you must obtain a user name and password that are authorized to access the SANnav server through the SANnav REST API.

To use the HTTPS protocol, a valid security certificate must be installed on the server before beginning REST operations. SANnav installs with a self-signed certificate. The SANnav certificate files are located in the `<install_home>/conf/nginx` directory on the SANnav server.

Session Authorization Key

To log in to a SANnav server, you must provide a valid SANnav user name and password through an authorization header in a POST login request. If the authentication is successful, SANnav returns the session ID in the body (for example: `"sessionId":"dd903934-f4d7-4eee-b05f-a7d2f48a733c"`). Subsequent SANnav REST API operations must include this session ID in the request authorization header. The client applications use this token to obtain further access to the server using the persistent connection. For more information, see Logging In.

Using SANnav REST API Session-Less Operation

SANnav REST API session-less operation allows you to provide authentication credentials directly for each REST API operation. Essentially, SANnav REST API session-less operation completes the login and the REST API operation as one complete request. SANnav REST API session-less operation does not create a session.

Keep the following in mind when using SANnav REST API session-less operation.

Before You Begin

Before you can use SANnav REST API session-less operation, you must convert a SANnav authorized user name and password (`User:Password`) to the Base64 authentication format to use in the authorization header. For example,

converting `User:Password` using the Base64 encoding results in `VXNlcjpQYXNzd29yZA==` . You must then prefix the base64 encoding with `Basic` (for example, `Basic VXNlcjpQYXNzd29yZA==`).

To use the HTTPS protocol, a valid security certificate must be installed on the server before beginning REST operations. SANnav installs with a self-signed certificate. The SANnav certificate files are located in the `<install_home>/conf/nginx` directory on the SANnav server.

Each time that you perform a SANnav REST API session-less operation request, several properties are checked for validation (license, password policy, role and area of responsibility (AOR) check, password expiry, and account locked) during user authentication. If any of these checks fails, the SANnav REST API session-less operation request fails.

NOTE
SAML 2.0 is not supported for External API.

Performing a Request Using SANnav REST API Session-Less Operation

You can use the Base64 authentication format only for SANnav REST API session-less operation. The following example shows a GET request using Base64 authentication.

`GET /external-api/v1/discovery/fabrics`

This example uses a GET request with Base64 authentication to retrieve information about the discovered fabrics in JSON format.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The Base64-encoded user credentials, which are the SANnav user name and password encoded using Base64. You must provide the Base64-encoded user credentials in the format `Basic <encoded_user_credentials>`. Example: `Basic VXNlcjpQYXNzd29yZA==`	string

Responses

HTTP Code	Description	Schema
200	All fabric details retrieved successfully.	FabricResponse
401	An unauthorized error occurred.	No Content
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

Wildcard Query Parameter Support

You can use the wildcard (*) character as input for an inventory search. The query string is sent in the URI after the question mark (?) delimiter and is usually in name=value pair formats.

The wildcard (*) search is supported only on the `/external-api/v1/inventory/search/` API. The wildcard (*) search is case-sensitive. A wildcard (*) can be entered using any of the following options in the search value:

- *abc
- ab*cd
- abc*
- *abcd*
- *

If no entry is present in the database, the search returns an empty list.

Object Type	Search Parameters	Additional Information
Switch	• switchWWN • serialnumber	This object type requires an additional optional parameter (parentId) to retrieve the correct data. The parentId refers to the switch WWN.
Switch port	• switchportWWN • switchportname • portfcid (FC address) • macaddress	—
Device	devicenodeWWN	—
Device port	deviceportwwn	—
Zone alias	• devicenodeWWN • deviceportwwn	—

Examples

This section provides a couple of basic examples for using the wildcard (*) search.

Retrieving a List of Switches

The following sample request and response use a GET request to perform a wildcard (*) search for a switch using the switch WWN.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/inventory/search/?
objecttype=switch&searchparam=switchWWN&value=<switch_WWN>
```

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/search/?objecttype=switch&searchparam=switchWWN&value=10:00*
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following and a “200 OK” status in the headers.

```
{
```

```
  "Searchquery": "10:00:*",
  "Inventory": [
    {
      "switch": "Vp58_f50",
      "searchresult": "10:00:00:01:11:E1:B1:11",
      "principleswitchwwn": "10:00:00:01:11:E1:B1:11",
      "switchipaddress": "0.0.0.0",
      "switchwwn": "10:00:00:01:11:E1:B1:11",
      "seedswitchwwn": "10:00:00:22:F2:F2:22:22",
      "vfid": "50",
      "fabric": "Untitled New Fabric 2"
    },
    {
      "switch": "Brocade6505",
      "searchresult": "10:00:00:33:F3:3A:33:EF",
      "principleswitchwwn": "10:00:00:33:F3:3A:33:EF",
      "switchipaddress": "10.10.10.29",
      "switchwwn": "10:00:00:33:F3:3A:33:EF",
      "seedswitchwwn": "10:00:00:33:F3:3A:33:EF",
      "vfid": "-1",
      "fabric": "MAPS"
    },
    {
      "switch": "Ap68_f50",
      "searchresult": "10:00:00:22:F2:F2:22:22",
      "principleswitchwwn": "10:00:00:04:44:E4:B4:44",
      "switchipaddress": "10.10.10.68",
      "switchwwn": "10:00:00:22:F2:F2:22:22",
      "seedswitchwwn": "10:00:00:22:F2:F2:22:22",
      "vfid": "50",
      "fabric": "Untitled New Fabric 2"
    },
  ]
}
```


Retrieving a List of Device Ports

The following sample request and response use a GET request to perform a wildcard (*) search for device ports using a device port WWN.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/inventory/search/?
objecttype=deviceport&searchparam=deviceportWWN&value=<switch_WWN>
```

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/search/?
objecttype=deviceport&searchparam=deviceportWWN&value=*14:18:00
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following and a “200 OK” status in the headers.


```
{
  "Searchquery": "*14:18:00",
  "Inventory": [
    {
      "switch": "SWNew11",
      "searchresult": "11:1D:11:1E:1E:14:18:00",
      "principleswitchwwn": "",
      "switchipaddress": "",
      "switchwwn": "10:00:10:EB:1A:11:11:54",
      "seedswitchwwn": "",
      "vfid": "",
      "fabric": "MAPS"
    }
  ]
}
```

Northbound Streaming

Northbound streaming provides support to securely stream performance and flow metrics from SANnav to an external Kafka cluster (the northbound server). Streaming gives you access to a large set of data from one or more managed fabrics that can be used to build customized reports or applications.

Raw SNMP and FOS streamed metrics that are received from the switch for performance metrics (PM) data and raw flow metrics that are received from switch streaming are streamed by SANnav to a northbound server. Configuration of the streaming interface is controlled using the REST API.

Before you configure northbound streaming, note the following prerequisites:

- You must have the Northbound Streaming privilege with read/write permission and the All Fabrics area of responsibility (AOR). Refer to the *Brocade SANnav Management Portal User Guide* for details.
- Historical data collection must be enabled on the SANnav server (it is enabled by default).
- The switch must be discovered with proper SNMPv3 credentials.
- SANnav must be registered on flow-enabled switches to receive flow data.
- For a northbound server environment with HTTPS access for the schema registry, the northbound Kafka cluster must be configured with the same public certificate as the northbound server.

Also note the following items:

- Only one northbound server is supported.
- Streaming is only through a secure channel (TLS).
- IPv6 is not supported. SANnav-to-northbound-server communication uses IPv4.

A heartbeat check is sent from SANnav to the northbound server every 2 seconds. Streaming is disabled if the heartbeat operation fails and resumes only after the next successful heartbeat response. Any data that is received by SANnav during the inactive period is discarded. The heartbeat check may fail in the following scenarios:

- The external Kafka broker is down.
- The external schema registry is down.
- A timeout occurred due to a network issue.
- The certificate expired.

Northbound streaming is registered as a listener for SANnav license update messages. If the SANnav license expires, northbound streaming stops. Northbound streaming resumes after you renew the SANnav license. However, note that SANnav discards any performance and flow data received during the expiration period and the data is not re-sent when northbound streaming resumes.

Stream Types

The following list details the supported stream types:

- 1: FCPORT (FC port metrics)
- 2: ETH/GIGE PORT (TE port and GigE port metrics)
- 3: EXTENSION (Extension tunnel and circuit metrics)
- 4: SWITCH (Switch performance metrics)
- 5: FLOW (Flow metrics: Gen 6 and Gen 7 switch flows)

For the FCPORT, ETH/GIGE PORT, EXTENSION, and SWITCH stream types, performance statistics data is streamed to the northbound server with a 5-minute granularity. For FLOW stream types, Gen 6 and Gen 7 switch flows are streamed at 5-minute and 6-hour granularity.

Platform Requirements for the Northbound Server

The northbound server Kafka requirements are the same as for the requirements for SANnav:

- Kafka version 6.0.1
- Confluent Platform for Schema Registry and Zookeeper version 6.0.1

Supported Topics

When you register the northbound server with SANnav, the following topics are automatically created if they do not already exist. Existing topics are not changed. If you want to change the partition count for each topic, you must do so before registering the northbound server.

Stream Type	Stream Description	Topic Name	Stream Interval	Recommended Partition Count
FCPORT	FC Port Statistics	perfmon_metrics_fcport	5 minutes	16
ETH/GIGE PORT	TE and GigE Port Statistics	perfmon_metrics_ethport	5 minutes	16
EXTENSION	Extension Tunnel Statistics	perfmon_metrics_extn_tunnel	5 minutes	16
	Extension Circuit Statistics	perfmon_metrics_extn_circuit	5 minutes	
SWITCH	Switch Performance Statistics	perfmon_metrics_switch	5 minutes	16
FLOW	Switch (Gen 6) Flow Statistics	flowmon_metrics_switch_flow	10 seconds, 5 minutes	16
	Switch (Gen 7) Flow Statistics		10 seconds, 5 minutes, 6 hours	

NOTE
The heartbeat_metrics_switch topic is used for the heartbeat check between SANnav and the northbound server. This topic is used for internal purposes and must not be altered or removed.

Stream Keys and Values

The following table lists the keys and values that are injected into the appropriate northbound topics for streaming. The Kafka message key is used by the northbound Kafka producer to determine the appropriate partition based on the hash of the key.

Table 1: Stream Keys and Values

Topic	Kafka Message Key	Kafka Message Value
perfmon_metrics_fcport	Port WWN Example: 20:14:50:EB:1A:0C:C8:4F	fibrechannel_port_statistics
perfmon_metrics_ethport	MAC Address Example: 00:27:F8:68:D1:90	ethernet_port_statistics
perfmon_metrics_extn_tunnel	Switch WWN + User Port Index Example: 10:00:00:05:33:e7:cf:8f_90	extension_tunnel_statistics
perfmon_metrics_extn_circuit	Switch WWN + User Port Index + Circuit Index Example: 10:00:00:05:33:e7:cf:8f_90_1	extension_circuit_statistics

Topic	Kafka Message Key	Kafka Message Value
perfmon_metrics_switch	Switch WWN Example: 10:00:00:05:33:e7:cf:8f	fibrechannel_switch_statistic
flowmon_metrics_switch_flow	null	fibrechannel_flow_statistics

For all supported streams, the Kafka key type now uses the Avro model. You must add schema to model the Key for each topic using Avro.

Example:

```
stream_key_external_schema.avsc{
   "namespace": "com.brocade.streaming",
   "name": "stream_key",
   "type": "record",
   "version": "1",
   "fields": [
      {"name": "key", "type": "string", "doc": "Stream Key. The value is different based on the Stream type."}
   ]
}
```

Schema

An Avro schema is defined and available for each stream type. The schema files are in the following location: `<install_home>/conf/northbound/streaming`

The following Avro schema models are available (file names are in parentheses):

- Extension circuit schema (`extn_circuit_external_schema.avsc`)
- Extension tunnel schema (`extn_tunnel_external_schema.avsc`)
- FC port schema (`fc_port_external_schema.avsc`)
- Gen 6 and Gen 7 flow schema (`fc_flow_external_schema.avsc`)
- Stream key schema (`stream_key_external_schema.avsc`)
- Switch schema (`switch_external_schema.avsc`)
- TE and GigE port schema (`eth_port_external_schema.avsc`)

NOTE

- Gen 6 supports VIT flows in 5-minute intervals.
- Gen 7 supports VITL flows in 10-second and 6-hour intervals.

Resources

A resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it.

A resource has a set of methods such as GET, POST, PUT, and DELETE are defined for it. Resources can be grouped into collections. Each collection is homogeneous (a collection contains only one type of resource) and unordered. Refer to the following table for a description of the methods.

Table 2: Supported SANnav REST API Methods

Method	Description
DELETE	Deletes the specified resource.
GET	Retrieves the representation of the resource (for example, “base, configuration”) including the metadata.
POST	Creates a new resource in the resource location identified by the URI specified in the request.
PUT	Replaces a resource in the resource location identified by the URI specified in the request.

Login and Logout

Use these API calls to log in and out of the REST API.

POST /external-api/v1/login

Description

Authenticates with the SANnav Management Portal server and retrieves the session ID to be used for subsequent REST API calls to the SANnav Management Portal server.

> **NOTE**
> Use this REST API call for session-based operation only.

Parameters

Type	Name	Description	Schema
Header	username (required)	The log-in user name using an external API.	string
Header	password (required)	The log-in password using an external API.	string (password)

Responses

HTTP Code	Description	Schema
200	The session ID returned for a successful login.	LoginResponse
401	An unauthorized error occurred.	No Content
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

POST /external-api/v1/logout/

Description

Logs you out of the SANnav Management Portal server.

NOTE
Use this REST API call for session-based operation only.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string

Responses

HTTP Code	Description	Schema
200	A successful logout.	No Content
401	An unauthorized error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

Discovery

Use these API calls to list discovered fabrics and fabric members.

GET /external-api/v1/discovery/fabrics

Description

Lists all discovered fabrics within the scope of the user's area of responsibility (AOR).

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string

Responses

HTTP Code	Description	Schema
200	All fabric details retrieved successfully.	FabricResponse

HTTP Code	Description	Schema
401	An unauthorized error occurred.	No Content
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

GET /external-api/v1/discovery/fabric-members/

Description

Lists all discovered fabrics within the scope of the user's area of responsibility (AOR).

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Query	principalSwitchWWN (required)	The principal switch WWN.	string

Responses

HTTP Code	Description	Schema
200	All switch details retrieved successfully.	FabricResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
402	A not found error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

FCR

Use this API call to view the FCR topology.

GET /external-api/v1/fcr/topology/

Description

Provides FCR topology information for various routing topologies such as edge-to-edge, backbone-to-edge, and edge-to-backbone, including the translate and front domain details.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string

Responses

HTTP Code	Description	Schema
200	Retrieved FCR topology information.	TopologyResponse
401	An unauthorized error occurred.	No Content
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

Fault

Use these API calls for Events.

POST /external-api/v2/fault/events/

Description

Lists the events that match the provided criteria.

Parameters

Type	Name	Description	Schema
Header	Authorization *(required)*	The session ID or the session-less API value (`Basic <base64_encoded_credentials>` .	string
Body	eventCriteria *(optional)*	The search criteria to retrieve a list of events.	FaultEventDetailsCriteria

Responses

HTTP Code	Description	Schema
200	An Events Response Object. A list of event counts for each severity group.	FaultEventsResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

POST /external-api/v1/fault/events/acknowledge

Description

A bulk acknowledgment for a list of events.

Parameters

Type	Name	Description	Schema
Header	Authorization *(required)*	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Body	acknowledgeEvents *(required)*	The filter criteria for the events to be acknowledged.	EventsAcknowledge

Responses

HTTP Code	Description	Schema
200	The acknowledgement details for the filtered events successfully updated.	SuccessResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

POST /external-api/v1/fault/events/unacknowledge

Description

Bulk unacknowledgment for a list of events.

Parameters

Type	Name	Description	Schema
Header	Authorization *(required)*	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Body	unacknowledgeEvents *(required)*	The filter criteria for the events to be unacknowledged.	EventsAcknowledge

Responses

HTTP Code	Description	Schema
200	The unacknowledgement details for the filtered events successfully updated.	SuccessResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

POST /external-api/v2/fault/events/forwarding/subscribe

Description

Creates a SANnav Management Portal event recipient.

Parameters

Type	Name	Description	Schema
Header	Authorization *(required)*	The session ID or the session-less API value (`Basic <base64_encoded_credentials>` .	string
Body	messageDetail *(required)*	The event recipient details.	EventRecipientDetails

Responses

HTTP Code	Description	Schema
200	Event recipient created.	SuccessResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
412	A precondition failed; occurs if recipient already exists.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

POST /external-api/v2/fault/events/forwarding/unsubscribe

Description

Deletes a SANnav Management Portal event recipient.

Parameters

Type	Name	Description	Schema
Header	Authorization *(required)*	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`).	string
Body	messageDetail *(required)*	The details of the event recipient to be deleted.	UnsubscribeForwardingRecipient

Responses

HTTP Code	Description	Schema
200	Event recipient deleted.	SuccessResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
412	A precondition failed; occurs if the recipient does not exist.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

Inventory

Use these API calls to view inventory.

GET /external-api/v1/inventory/search/

Description

Retrieves inventory details based on the search criteria.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Query	objecttype (required)	The object type.	enum (Switch, SwitchPort, Device, DevicePort, ZoneAlias)
Query	searchparam (required)	The search parameter.	enum (switchWWN, serialnumber, switchportWWN, switchportname, portfcid, macaddress, deviceNodeWWN, deviceportWWN)
Query	value (required)	The value.	string
Query	parentId (optional)	The parentId is the switch WWN. This is the parameter for the switch port.	string

Responses

HTTP Code	Description	Schema
200	The inventory details successfully retrieved.	InventoryOutputResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
404	A not found error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

GET /external-api/v1/inventory/chassis/

Description

Retrieves the inventory details for all chassis based on the search criteria.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>` .	string
Query	Scope (optional)	The scope to search the chassis.	enum (physicalSwitchWWN, physicalSwitcName
Query	Value (optional)	The search value.	string
Query	basicOnly (optional)	Whether to retrieve only basic attributes. 1 = yes; 0 = no. If no, returns additional attributes. The default is 1.	integer

Responses

HTTP Code	Description	Schema
200	The switch inventory details successfully retrieved.	InventoryChassisResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
404	A not found error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

GET /external-api/v1/inventory/deviceports/

Description

Retrieves the inventory details for all device ports based on the search criteria.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>` .	string
Query	Type (optional)	Device type to search; blank for both host and storage.	enum (host, storage)
Query	Scope (optional)	The scope to search device ports. The activeZoneNameExact search option is case-sensitive, and returns only objects that match the query value exactly.	enum (enclosureName, fabricName, zoneAlias, portWWN, portOrNodeWWN, activeZoneName, activeZoneNameExact)
Query	Value (optional)	The search value.	string
Query	startIndex (optional)	The start index for pagination. The default is zero.	integer
Query	numOfRecords (optional)	The number of records to retrieve. The maximum is 5000 or what is defined in the server properties (`server.properties`).	integer
Query	basicOnly (optional)	Whether to retrieve only basic attributes. 1 = yes; 0 = no. If no, returns additional attributes. The default is 1.	integer

Responses

HTTP Code	Description	Schema
200	The switch inventory details successfully retrieved.	InventoryDevicePortsResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
404	A not found error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

GET /external-api/v1/inventory/enclosures/

Description

Retrieves the inventory details for all enclosures based on the search criteria.

NOTE
This REST API does not honor areas of responsibility (AORs).

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Query	deviceType (optional)	Device type to search; blank for both host and storage.	enum (host, storage)
Query	Scope (optional)	The scope to search enclosures.	enum (enclosureName)
Query	Value (optional)	The search value.	string
Query	startIndex (optional)	The start index for pagination. The default is zero.	integer
Query	numOfRecords (optional)	The number of records to retrieve. The maximum is 5000 or what is defined in the server properties.	integer
Query	basicOnly (optional)	Whether to retrieve only basic attributes. 1 = yes; 0 = no. If no, returns additional attributes. The default is 1.	integer

Responses

HTTP Code	Description	Schema
200	The switch inventory details successfully retrieved.	InventoryEnclosuresResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
404	A not found error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

POST /external-api/v1/inventory/enclosures/mapping

Description

NOTE
In SANnav 2.3.0 or later, to apply offline mapping when the device port comes online, you must enable an auto enclosure for the Host or Storage (refer to the *Brocade SANnav Management Portal User Guide*).

Maps a device port to an enclosure based on the following behavior:

- If the device port, WWN case-insensitive comparison, exists in SANnav; however, the enclosure does not exist in SANnav, create the enclosure object in SANnav and map the device port to the enclosure.
- If the device port is part of an unmonitored fabric or not in your AOR, the device port is skipped. However, the remaining device ports are mapped. The device ports that belong to an unmonitored switch (Connected Switch), but a monitored fabric are mapped.
- If the mapping exists in SANnav and matches the SANnav data, skip and continue to the next option.
- If the mapping exists in SANnav, however it does not match the input data and the override is set to false (0), error and continue (default) to the next option. However, if the override is set to true (1), then override the mapping and continue to the next option.
- If the WWN and the mapping exist in SANnav, input the WWN without an enclosure name which deletes the mapping.
- If the WWN does not exist in SANnav (for example, the device is not online yet), persist and create the mapping when the device is online.

Parameters

Type	Name	Description	Schema
Header	Authorization *required*	The session ID or the session-less API value (`Basic <base64_encoded_credentials>` .	string
Query	mappings *required*	Maps a device port to an enclosure.	EnclosureMappingInput

Responses

HTTP Code	Description	Schema
200	The device port to the enclosure successfully mapped.	InventoryEnclosureMappingsResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
404	A not found error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

GET /external-api/v1/inventory/fabrics/

Description

Retrieves the inventory details for all fabrics based on the search criteria.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Query	Scope (optional)	The scope to search fabrics.	enum (fabricName)
Query	Value (optional)	The search value.	string
Query	basicOnly (optional)	Whether to retrieve only basic attributes. 1 = yes; 0 = no. If no, returns additional attributes. The default is 1.	integer

Responses

HTTP Code	Description	Schema
200	The fabric inventory details successfully retrieved.	InventoryFabricsResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
404	A not found error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

GET /external-api/v1/inventory/switches/

Description

Retrieves the inventory details for all switches based on the search criteria.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Query	Scope (optional)	The scope to search switches.	enum (switchWWN, switcName, ipAddress, fabricName)
Query	Value (optional)	The search value.	string
Query	startIndex (optional)	The start index for pagination. The default is zero.	integer

Type	Name	Description	Schema
Query	numOfRecords (optional)	The number of records to retrieve. The maximum is 5000 or what is defined in the server properties.	integer
Query	basicOnly (optional)	Whether to retrieve only basic attributes. 1 = yes; 0 = no. If no, returns additional attributes. The default is 1.	integer

Responses

HTTP Code	Description	Schema
200	The switch inventory details successfully retrieved.	InventorySwitchesResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
404	A not found error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

GET /external-api/v1/inventory/switchports/

Description

Retrieves the inventory details for all switch ports based on the search criteria.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Query	Scope (optional)	The scope to search switch device ports.	enum (switchWWN, switchName, portWWN, portMacAddress, portName, fabricName, ipAddress, physicalSwitchWWN)
Query	Value (optional)	The search value.	string
Query	startIndex (optional)	The start index for pagination. The default is zero.	integer
Query	numOfRecords (optional)	The number of records to retrieve. The maximum is 5000 or what is defined in the server properties.	integer

Type	Name	Description	Schema
Query	basicOnly (optional)	Whether to retrieve only basic attributes. 1 = yes; 0 = no. If no, returns additional attributes. The default is 1.	integer

Responses

HTTP Code	Description	Schema
200	The switch inventory details successfully retrieved.	InventorySwitchPortsResponse
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No Content
404	A not found error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

Stream

Use these API calls to manage northbound streams.

GET /external-api/v1/stream

Description

Retrieves the supported streams and possible stream states. Returns a StreamMetaData, which contains a list of stream types and a list of possible stream states.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string

Responses

HTTP Code	Description	Schema
200	Successfully retrieved the northbound server details.	NorthboundServer array
401	Unauthorized. The request parameter is not valid. The response body contains additional information.	ErrorResponse

HTTP Code	Description	Schema
500	Internal error. The server encountered an error while handling the valid request. The response body contains additional information.	ErrorResponse

Accept

application/json

Content Type

application/json

GET /external-api/v1/stream/servers

Description

Retrieves all registered northbound servers. Returns a list of northbound server details with Kafka Broker URL, Schema Registry URL, and Encoded CA Public Certificate in Base64 format. Also includes a list of stream details with stream ID and state.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string

Responses

HTTP Code	Description	Schema
200	Successfully retrieved the northbound server details.	NorthboundServer array
401	Unauthorized. The request parameter is not valid. The response body contains additional information.	ErrorResponse
500	Internal error. The server encountered an error while handling the valid request. The response body contains additional information.	ErrorResponse

Accept

application/json

Content Type

application/json

POST /external-api/v1/stream/servers

Description

Creates a northbound server. The request should include the northbound server name, the Kafka broker URL, schema registry URL, and encoded CA public certificate in Base64 format. Returns the northbound server details received in the request with the server ID and a list of stream details.

Parameters

Type	Name	Description	Schema
Header	Authorization *(required)*	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Body	northboundServerConfig *(required)*	The northbound server configuration.	NorthboundServerConfig

Responses

HTTP Code	Description	Schema
201	Created. The northbound server details are in the response body.	NorthboundServer
400	Bad request. The NorthboundServerConfig request is not valid. The response body contains additional information.	ErrorResponse
401	Unauthorized. The authorization request parameter is not valid. The response body contains additional information.	ErrorResponse
500	Internal error. The server encountered an error while handling the valid request. The response body contains additional information.	ErrorResponse

Accept

application/json

Content Type

application/json

GET /external-api/v1/stream/servers/{serverId}

Description

Retrieves the northbound server details. Returns the northbound server details with the Kafka broker URL, schema registry URL, and encoded CA public certificate in Base64 format. Also includes a list of stream details with the stream ID and state.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Path	serverId (required)	The server identifier.	integer

Responses

HTTP Code	Description	Schema
200	Ok. The requested northbound server data is in the response body.	NorthboundServer
401	Unauthorized. The authorization request parameter is not valid. The response body contains additional information.	ErrorResponse
404	Not found. Invalid borthbound server ID. The response body contains additional information.	ErrorResponse
500	Internal error. The server encountered an error while handling the valid request. The response body contains additional information.	ErrorResponse

Accept

application/json

Content Type

application/json

DELETE /external-api/v1/stream/servers/{serverId}

Description

Stops all active streaming for the specified server and then removes the northbound server.

To remove the northbound server, you must have the Northbound Streaming privilege with read-write permission.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Path	serverId (required)	The server identifier.	integer

Responses

HTTP Code	Description	Schema
200	Ok. The requested northbound server is removed.	NorthboundServer
401	Unauthorized. The authorization request parameter is not valid. The response body contains additional information.	ErrorResponse
404	Not found. Invalid northbound server ID. The response body contains additional information.	ErrorResponse
500	Internal error. The server encountered an error while handling the valid request. The response body contains additional information.	ErrorResponse

Accept

application/json

Content Type

application/json

PUT /external-api/v1/stream/servers/{serverId}/streams

Description

Enables or disables a stream. Enabling a stream starts streaming to the requested northbound server. Disabling a stream stops streaming to the requested northbound server.

To enable or disable a stream, you must have the Northbound Streaming privilege with read-write permission.

Parameters

Type	Name	Description	Schema
Header	Authorization *required*	The session ID or the session-less API value (`Basic <base64_encoded_credentials>`.	string
Path	serverId *required*	The server identifier.	integer
Body	streamConfig *required*	The stream configuration.	StreamConfig

Responses

HTTP Code	Description	Schema
200	The stream state successfully updated.	NorthboundServer
400	A bad request error occurred.	No content.
401	An unauthorized error occurred.	ErrorResponse
404	A not found error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

Health Summary

Use this API call to a view the health summary.

POST /external-api/v1/healthsummary/healthsummarydetails/

Description

Retrieves the health summary for the requested entity.

Parameters

Type	Name	Description	Schema
Header	Authorization (required)	The session ID.	string
Body	requestEntity (required)	The health summary details.	RequestHealthEntity

Responses

HTTP Code	Description	Schema
200	Successfully retrieved the health summary.	HealthSummaryDetails
400	A bad request error occurred.	ErrorResponse
401	An unauthorized error occurred.	No content
404	A not found error occurred.	ErrorResponse
500	An unexpected error occurred.	ErrorResponse

Accept

application/json

Content Type

application/json

Definitions

The SANnav REST API supports many common definitions that are used by multiple APIs. Additional details about these common definitions can be found in a model file (`external-api-model.yaml`).

AdditionalCoreSwitchDetailsInfo

Name	Description	Schema
actCpPriFirmwareVersion *optional*	The active CP primary firmware version.	string
actCpSecFirmwareVersion *optional*	The active CP secondary firmware version.	string
callhomeEnabled *optional*	Whether the call home feature is enabled. 1: Call home is enabled. 0: Call home is disabled.	integer (int32)
creationTime *optional*	The core switch record creation time. This time is when the initial discovery occurred.	string (date-time)
egmCapable *optional*	Whether the EGM license is supported. 1: The EGM license is supported. 0: The EGM license is not supported.	integer (int32)
fcCertificate *optional*	The FC certificate number.	integer (int32)
fcIp *optional*	The FC IP address.	string
fcMask *optional*	The FC IP address Ethernet mask.	string
lastScanTime *optional*	The last scan time. The last time the switch was polled.	string (date-time)
lastUpdateTime *optional*	The last update time. The last update to the record.	string (date-time)
maintenanceEndTime *optional*	The maintenance end time.	string (date-time)
maintenanceMode *optional*	The maintenance mode of the switch. 0: Off 1: On 2: Scheduled	integer (int32)
maintenanceStartTime *optional*	The maintenance start time.	string (date-time)
mapsEnabled *optional*	Whether MAPS is enabled. 1: MAPS is enabled. 0: MAPS is disabled.	integer (int32)
nicProfileId *optional*	The entry in the NIcProfile table that has the SANnav IP address that is used for the syslog and SNMP reciepients.	integer (int32)
snmpRegistered *optional*	Whether the switch is registered for sending SNMP traps. 1: The switch is registered. 0: The switch is not registered.	integer (int32)
stbyCpPriFirmwareVersion *optional*	The standby CP primary firmware version.	string
stbyCpSecFirmwareVersion *optional*	The standby CP secondary firmware version.	string

Name	Description	Schema
syslogRegistered *optional*	Whether the switch is registered for sending syslog. 1: The switch is registered. 0: The switch is not registered.	integer (int32)
unreachableTime *optional*	The time when the switch became unreachable.	string (date-time)
vfEnabled *optional*	Whether VF is enabled. 1: VF is enabled. 0: VF is disabled.	integer (int32)
vfSupported *optional*	Whether VF is supported. 1: VF is supported. 0: VF is not supported.	integer (int32)

AdditionalDeviceEnclosureInfo

Name	Description	Schema
createdBy *optional*	The module that created this enclosure. • 0: Manual • 1: HBA • 2: VM • 3: AUTO_ENCLOSURE	integer (int32)
creationTime *optional*	The time when the enclosure was created.	string (date-time)
hcmAgentVersion *optional*	The version of the HCM agent running on the host.	string
lastUpdateModule *optional*	The module that updated the host information.	integer (int32)
lastUpdateTime *optional*	The last update to the record time.	string (date-time)
managedBy *optional*	The module that managed the enclosure. 1: Manual (user created, not a managed condition) 2: Host Adapter 3: VMM 4: Host Adapter and VMM The default is 1 (manual).	integer (int32)
osVersion *optional*	The operating system version for the enclosure.	string
queueDepth *optional*	The queue depth controls the FCP exchange resource allocation. Valid values are 0 to 254. The default value is 32.	integer (int32)
syslogRegistered *optional*	Whether syslog is enabled.	integer (int32)
virtualization *optional*	If this enclosure is a host, this column indicates whether the host is running a virtualization hypervisor. 0 = unknown 1 = no supported hypervisor present 2 = VMware ESX 3 = Microsoft Hyper-V	integer (int32)

AdditionalDevicePortInfo

Name	Description	Schema
agNPortWwn *optional*	If the device is connected with an AG, the N_Port WWN of the AG, which is connected to a switch, is populated from the N2F and N2WWN map.	string
agNodeWwn *optional*	If the device is connected with an AG, the AG switch WWN is populated. This is not a null field, and the default value is empty.	string
cos *optional*	The port class of service.	string
creationTime *optional*	The time when the device port entry was created.	string (date-time)
deviceEnclosureIpAddress *optional*	The IP address of the enclosure.	string
deviceEnclosureUddTags *optional*	The user-defined enclosure tag.	string
deviceEnclosureUddType *optional*	The enclosure type (HBA host or storage array).	string
fc4Type *optional*	The port FC4 type (for example FCP).	string
fdmiDriverVersion *optional*	The driver version of the device available via FDMI.	string
fdmiFirmwareVersion *optional*	The firmware version of the device available via FDMI.	string
fdmiHardwareVersion *optional*	The hardware version of the device available via FDMI.	string
fdmiManufacturer *optional*	The manufacturer of the device available via FDMI.	string
fdmiModel *optional*	The model of the device available via FDMI.	string
fdmiModelDescription *optional*	The model description of the device available via FDMI.	string
fdmiNodeName *optional*	The node name of the device available via FDMI (for example 20:00:00:05:1e:7c:64:94).	string
fdmiOsDetails *optional*	The OS name and version of the device available via FDMI.	string
fdmiSerialNumber *optional*	The serial number of the device available via FDMI.	string
fdmiSupportedSpeeds *optional*	The supported speeds of the device available via FDMI (for example, speed-2-gfc, speed-4-gfc, speed-8-gfc, speed-16-gfc).	string
flag *optional*	The FICON device property (for example, 0x10 [hex]).	string
guid *optional*	The unique ID of the device port.	string
loggedToAg *optional*	Whether the device is connected with an AG. This is not a null field, and the default value is 0 (device is not connected to AG).	integer (int32)
logicalPortWwn *optional*	The logical port WWN of the F_Port trunk group.	string
managed *optional*	Whether the fabric is managed.	integer (int32)
manufacturer *optional*	The FICON manufacturer of the device, typically IBM.	string
manufacturerPlant *optional*	The plant number where the device was manufactured.	string
modelNumber *optional*	The FICON device model number, such as S18.	string
monitored *optional*	Whether the connected switch is monitored.	integer (int32)
npvPhysical *optional*	Whether the port is a physical device port or a logical NPIV port.	integer (int32)
params *optional*	The FICON device property string (for example, a valid channel port).	string
portIndex *optional*	The connected switch port index.	integer (int32)
portState *optional*	The connected switch port state, online or offline.	string
previoursMissingState *optional*	The previous missing state of this device. Default value is 0.	integer (int32)
principalWwn *optional*	The principal switch WWN.	string

Name	Description	Schema
sequenceNumber *optional*	The FICON sequence number.	string
switchGuid *optional*	The connected switch GUID.	string
switchPortGuid *optional*	The connected switch port GUID.	string
switchPortPortId *optional*	The connected switch port FC address.	string
switchPortUddTags *optional*	The connected switch port tags.	string
tag *optional*	The FICON device property (for example, 809a or 809b).	string
trusted *optional*	Whether the device port is trusted.	integer (int32)
typeNumber *optional*	The FICON type number.	string
userPortNumber *optional*	The user port number of the connected switch port.	integer (int32)

AdditionalFabricInfo

Name	Description	Schema
adEnvirnment *optional*	The AD environment of the fabric. 1 = There is a user-defined AD in the fabric.	integer (int32)
creationTime *optional*	The creation time of the record.	string (date-time)
guid *optional*	A globally unique identifier for an individual fabric.	string
lastScanTime *optional*	The last time the fabric was scanned for changes.	string (date-time)
lastUpdateTime *optional*	The last time the record was updated.	string (date-time)
managementState *optional*	A bit map that indicates the various management states of the fabric. The default value is 0.	integer (int32)
physicalSwitchGuid *optional*	The world-wide name of the chassis or switch.	string
sanId *optional*	The meta SAN DB identifier.	integer (int32)
seedSwitchId *optional*	The seed switch DB identifier.	integer (int32)
switchGuid *optional*	A globally unique identifier for the switch.	string

AdditionalSwitchInfo

Name	Description	Schema
activeCpPrimaryFirmwareVersion *optional*	The active CP primary firmware version.	string
activeCpSecondaryFirmwareVersion *optional*	The active CP secondary firmware version.	string
adCapable *optional*	The capability for the admin domain. Valid values are: 1: Capable. 0: Not capable.	integer (int32)
autoSnmp *optional*	Whether automatic SNMP configuration is enabled for the switch. Valid values are: 0: Manual. 1: Automatic.	integer (int32)
bound *optional*	Whether the SANnav server is bound to the switch.	integer (int32)

Name	Description	Schema
boundServerIpAddress *optional*	The IP address of the SANnav server that is bound to the switch. Displays as empty or 0.0.0.0 if no SANnav server is bound to the switch.	string
callhomeEnabled *optional*	Whether the call home feature is enabled. Valid values are: 1: Enabled. 0: Disabled.	integer (int32)
chassisType *optional*	The type of chassis of the switch. Valid values are: 0: Fixed-port switch. 1: Director (modular chassis).	integer (int32)
creationTime *optional*	The creation time for the record.	string (date-time)
cryptoCapable *optional*	Whether the switch is crypto capable. Valid values are: 1: The switch is crypto capable. 0: The switch is not crypto capable.	integer (int32)
defaultLogicalSwitch *optional*	Whether the virtual switch is a default logical switch. Valid values are: 1: True. 0: False.	integer (int32)
dynamicLoadSharing *optional*	Whether the switch supports the dynamic load sharing capability. Valid values are: 1: The switch is capable. 0: The switch is not capable.	integer (int32)
edgeHoldTime *optional*	The timeout, in milleseconds, configured on the switch to discard blocked frames within an F_ Port to reduce the likelihood of back pressure into the fabric.	integer (int32)
egmCapable *optional*	Whether the EGM license is supported. Valid values are: 1: Supported. 0: Not supported.	integer (int32)
embedded *optional*	Whether the switch is in embedded mode. Valid values are: 1: Embedded. 0: Not embedded.	integer (int32)
enforceEula *optional*	Whether the EULA is enforced by the firmware. Valid values are: 1: Enforced. 0: Not enforced.	integer (int32)
eosStatus *optional*	The end of support (EOS) status of a switch (chassis). Valid values are: –1: The EOS status is not verified. 0: The switch is supported (switch has not reached EOS or the customer added a license to extend the EOS period). 1: The switch is no longer supported because it has reached end of support.	integer (int32)
ethernetMask *optional*	The Ethernet mask of the core switch IP address.	string
fabricGuid *optional*	The unique fabric ID.	string
fcCertificate *optional*	The FC certificate number.	integer (int32)
fcIp *optional*	The FC IP address.	string
fcMask *optional*	The Ethernet mask of the FC IP address.	string
fcipCapable *optional*	The switch capability for FCIP support. Valid values are: 1: The switch is capable. 0: The switch is not capable.	integer (int32)
fcipCircuitCapable *optional*	The switch capability for FCIP circuit creation. Valid values are: 1: The switch is capable. 0: The switch is not capable.	integer (int32)
fcipLicensed *optional*	Whether FCIP Advanced Extension Licensing is available. Valid values are: 1 Licensed. 0: Not licensed. –1: Not supported.	integer (int32)
fcoeCapable *optional*	The switch capability for FCOE support. Valid values are: 1: The switch is capable. 0: The switch is not capable.	integer (int32)
fcoeLoginEnabled *optional*	The FCoE login management status of the switch.	integer (int32)
fcrCapable *optional*	The switch capability for FCR support. Valid values are: 1: The switch is capable. 0: The switch is not capable.	integer (int32)
fcsRole *optional*	The FCS role of the switch. This is applicable only when the FCS policy is on.	string

Name	Description	Schema
featureEnabled *optional*	A bit mask that indicates the active or enabled features. Each bit represents a feature (such as lossless).	integer (int32)
featureSupported *optional*	The supported features as a bit mask.	integer (int32)
fmsMode *optional*	The FMS mode in a FICON environment.	integer (int32)
guid *optional*	The unique ID for an individual switch.	string
hifEnabled *optional*	Whether high integrity fabric (HIF) mode is enabled. Valid values are: 1: Enabled. 0: Disabled.	integer (int32)
inOrderDelivery *optional*	Whether in-order delivery is enabled. Valid values are: 1: Enabled. 0: Disabled.	integer (int32)
insistentDidMode *optional*	Whether persistent domain ID is enabled on the switch. Valid values are: 1: Enabled. 0: Disabled.	integer (int32)
interopMode *optional*	The interop mode for the switch. Valid values are: 0: Native. 3: Open fabric.	integer (int32)
l2Capable *optional*	The switch capability for L2 support. Valid values are: 1: The switch is capable. 0: The switch is not capable.	integer (int32)
l3Capable *optional*	The switch capability for L3 support. Valid values are: 1: The switch is capable. 0: The switch is not capable.	integer (int32)
lastPortMembershipChange *optional*	The timestamp of the last port membership update.	integer (int64)
lastScanTime *optional*	The last time the switch was contacted for an update.	string (date-time)
lastUpdateTime *optional*	The last update time of the record.	string (date-time)
lfEnabled *optional*	Whether Logical Fabric is enabled for a virtual switch. Valid values are: 1: Enabled. 0: Disabled.	integer (int32)
maintenanceEndTime *optional*	The maintenance end time.	string (date-time)
maintenanceMode *optional*	The maintenance mode of the switch. Valid values are: 0: Off. 1: On. 2: Scheduled.	integer (int32)
maintenanceStartTime *optional*	The maintenance start time.	string (date-time)
managementState *optional*	The management state of the switch. A bit mask that indicates the manageability state of the switch, such as switch not reachable, credentials invalid, and not ready for management. A 0 value indicates that the management state is okay.	integer (int64)
managingServerIpAddress *optional*	The IP address (IPv4 or IPv6 format) of the switch.	string
mapsEnabled *optional*	Whether MAPS is enabled. Valid values are: 1: Enabled. 0: Disabled.	integer (int32)
mapsEnabledActions *optional*	A bit mask of MAPS actions enabled on the switch. Valid values are: 0: None. 1: Raslog. 2: SNMP. 4: Email. 8: Fence Port. 16: SW Down. 32: SW Marginal.	integer (int32)
maxFcipCircuits *optional*	The maximum number of circuits that can be created in the switch. Displays a –1 if it is not supported.	integer (int32)
maxFcipTunnels *optional*	The maximum number of tunnels that can be created in the switch. Displays a –1 if it is not supported.	integer (int32)
maxNumOfBlades *optional*	The maximum number of blades that can be present in a chassis.	integer (int32)
maxVirtualSwitches *optional*	The maximum number of logical switches supported.	integer
missingTime *optional*	The switch is missing from the fabric. This value is null if the member is entrusted.	string (date-time)
multiCpCapable *optional*	Whether the switch is capable of multiple CPs. 0: A single CP. 1: Multiple CPs.	integer (int32)

Name	Description	Schema
natPrivateIpAddress *optional*	The IP address (IPv4 or IPv6 format) of the switch.	string
nicProfileId *optional*	The IP address of the SANnav server configured as the syslog and SNMP recipient.	integer (int32)
numVirtualSwitches *optional*	The number of logical switches on the switch.	integer
partition *optional*	Whether partitions are supported in the switch.	integer (int32)
physicalSwitchGuid *optional*	A unique ID for an individual switch.	string
previoursOperationalStatus *optional*	The operational status of the switch.	string
rnidSequenceNumber *optional*	The sequence number of the switch.	string
snmpInformEnabled *optional*	Whether SNMP informs are enabled. Valid values are: 1: Enabled. 0: Disabled.	integer (int32)
snmpRegistered *optional*	Whether the switch is registered to send SNMP traps. Valid values are: 1: Registered. 0: Not registered.	integer (int32)
standbyCpPrimaryFirmwareVersion *optional*	The primary firmware version for the standby CP.	string
standbyCpSecondaryFirmwareVersion *optional*	The secondary firmware version for the standby CP.	string
subType *optional*	The subtype of the switch. Valid values are: 1: DCX+ and DCX-4S +. 0: All other subtypes.	string
syslogRegistered *optional*	Whether the switch is registered to send syslog. Valid values are: 1: Registered. 0: Not registered.	integer (int32)
unreachableTime *optional*	The time when the switch became unreachable.	string (date-time)
userDefinedValue1 *optional*	A user-defined value.	string
userDefinedValue2 *optional*	A user-defined value.	string
userDefinedValue3 *optional*	A user-defined value.	string
userIpAddress *optional*	The IP address of the physical switch.	string
vendor *optional*	The product vendor.	string
vendorPartNumber *optional*	The vendor SKU number.	string
vendorVersion *optional*	The vendor version.	string
vfEnabled *optional*	Whether VF is enabled. Valid values are: 1: Enabled. 0: Disabled.	integer (int32)
vfSupported *optional*	Whether VF is supported. Valid values are: 1: Supported. 0: Not supported.	integer (int32)

AdditionalSwitchPortInfo

Name	Description	Schema
actualDistance *optional*	The actual distance of the link.	integer (int32)
areaId *optional*	The area number to which the port is assigned.	integer (int32)
bufferAllocated *optional*	The number of B2B credits allocated to the port.	integer (int32)
desiredCredits *optional*	The number of B2B credits desired by the port.	integer (int32)
domainId *optional*	The domain ID of the switch.	integer (int32)
eportDisabled *optional*	Whether the E_Port is disabled.	integer (int32)
estimatedDistance *optional*	The estimated distance of the link.	integer (int32)

Name	Description	Schema
extType *optional*	The extended type of the port (such as Mirror-Port).	string
fabricGuid *optional*	The globally unique identifier (GUID) of the fabric.	string
fabricId *optional*	The unique fabric identifier.	integer (int32)
fabricMonitored *optional*	Whether the fabric is monitored or unmonitored. 0: Unmonitored 1: Monitored	integer (int32)
fcFastWriteEnabled *optional*	Whether FC Fast Write is enabled.	integer (int32)
fcipTunnelUp *optional*	Whether the configured FCIP tunnel is up.	integer (int32)
fcrFabricId *optional*	The FCR fabric ID. Applicable if the port is configured as an EX_Port.	integer (int32)
fcrInteropMode *optional*	The interop mode of the FCR fabric. Applicable if the port is an EX_Port.	integer (int32)
featuresActive *optional*	A bit mask that indicates which features are active.	integer (int32)
featuresEnabled *optional*	A bit mask that indicates which features are enabled.	integer (int32)
featuresSupported *optional*	The features supported as a bit mask at the port level.	integer (int32)
firmwareVersion *optional*	The firmware version.	string
fullType *optional*	The full type of the port (such as U_Port, F_Port).	string
guid *optional*	The unique ID for an individual port.	string
interopMode *optional*	The interop mode for the switch. 0: native 3: open fabric	integer (int32)
islRrdyEnabled *optional*	Whether ISL receiver ready is enabled.	integer (int32)
kind *optional*	The port kind from the NVP portKind.	string
lagName *optional*	The LAG name.	string
latencyDetectSupported *optional*	Whether the port supports latency detection. 1: true 0: false	integer (int32)
longDistanceSetting *optional*	Whether the long-distance setting is enabled on the switch.	integer (int32)
managementState *optional*	The management state of the switch. This is a bit mask that indicates the switches manageability state, such as switch not reachable, credentials invalid, not ready for manangement. 0: The management state is okay.	integer (int64)
physicalSwitchGuid *optional*	The GUID for an individual switch.	string
physicalSwitchId *optional*	The physical switch identifier.	integer (int32)
portBitMask *optional*	The F_Port trunk bit mask value.	integer (int32)
portCapabilities *optional*	A list of capabilities for this port specified as a bit mask. Each bit represents the capability such as FEC, encryption and compression, NPIV.	integer (int32)

Name	Description	Schema
portCommissionState *optional*	Whether port decommission or recommission is in progress or completed. 0: None 1: Decommission In Progress 2: Decommissioned 3: Recommission In Progress 4: Recommissioned If the decommission is performed using other tools, the state is 0 (None).	integer (int32)
prohibitPortCount *optional*	The count of prohibited ports.	integer (int32)
prohibitPortNumbers *optional*	The ports prohibited with the current port as configured in the allow/prohibit matrix (PDCM).	string
protocol *optional*	The protocol used by the port (such as, FC, FCIP).	string
qosCapable *optional*	Whether the port is QoS capable.	integer (int32)
qosEnabled *optional*	Whether the port is QoS enabled.	integer (int32)
qsfpMode *optional*	The QSFP mode for ports that have QSFP. Valid values are Breakout or Parallel.	string
qspfPrimaryPort *optional*	Whether the port is the QSFP primary port. 1: It is the QSFP primary port. 0: It is not the QSFP primary port. This parameter is applicable only for DD SFP and 50G SFP.	integer (int32)
rateLimited *optional*	Whether rate limiting is enabled on the port.	integer (int32)
remoteSwitchGuid *optional*	A unique identifier.	string
remoteSwitchPortGuid *optional*	A unique identifier.	string
routingPolicy *optional*	The routing policy (APT-advanced performance tuning) configured in the switch. The routing policy is not applicable for FOS AG, NOS, or mEOS switches. For non-AG FOS switches, the unknown value is set as the initial value. Once asset collection is successful, the corresponding value is fetched from the switch. Valid values: –1: Not Applicable 0: Unknown 1: Port Based Routing 2: Device Based Routing 3: Exchange Based Routing	integer (int32)
switchGuid *optional*	A unique ID for the switch.	string
switchModel *optional*	The switch model type.	integer (int32)
switchModelNumber *optional*	The model number of the switch (for example, Brocade 8000).	string
switchRole *optional*	The role of the switch (such as primary or subordinate).	string
switchSubType *optional*	The subtype of the switch. Valid values are: 1: DCX+ and DCX-4S +. 0: All other sub types.	string
tunnelConfigured *optional*	Whether the port has an FCIP tunnel configured.	integer (int32)
userDefinedValue1 *optional*	The user-defined value used for annotation.	string
userDefinedValue2 *optional*	The user-defined value used for annotation.	string
userDefinedValue3 *optional*	The user-defined value used for annotation.	string
userPortNumber *optional*	The user port number. A unique port number in a chassis.	integer (int32)

Name	Description	Schema
virtualSwitchId *optional*	The DB ID of the virtual switch to which this port belongs.	integer
xislPortList *optional*	The list of XISL ports associated with the current logical port. This field is applicable only for logical ports created for LISLs. This field is blank for nonlogical ports.	string

BackboneSwitch

Name	Description	Schema
backboneIPAddress *optional*	The router IP address.	string
backboneSwitchWwn *optional*	The backbone switch WWN.	string
edgeFabric *optional*	A list of edge fabrics.	<EdgeFabric> array

BulkConfig

Name	Description	Schema
addTags *optional*	Add new tags.	string
ids *optional*	The identifiers.	<string> array
removeAllTags *optional*	Remove all existing tags.	boolean
removeTags *optional*	Remove tags.	string

CategoryCriteria

Name	Description	Schema
correlation_event *optional*	The selected severity for a correlation event. Values include Emergency, Alert, Critical, Error, Warning, and Info.	<string> array
link_incident_event *optional*	The selected severity for a link incident event.	<string> array
management_server_event *optional*	The selected severity for a management server event.	<string> array
product_audit_event *optional*	The selected severity for a product audit event.	<string> array
product_event *optional*	The selected severity for a product event.	<string> array
product_status_event *optional*	The selected severity for a product status event.	<string> array
security_event *optional*	The selected severity for a security event.	<string> array
user_action_event *optional*	The selected severity for a user action event.	<string> array

ConflictDeviceObject

Name	Description	Schema
id *optional*	The identifier for the conflict device.	integer
wwn *optional*	The world-wide name for the conflict device.	string
currentDevice *optional*	The current device.	string
currentDeviceId *optional*	The identifier for the current device.	integer
importedDevice *optional*	The imported device.	string
importedDeviceId *optional*	The identifier for the imported device.	integer
type *optional*	The conflict device type.	string
issue *optional*	The issue.	string
userAction *optional*	The user action.	string

ConflictDeviceObjectResponse

Name	Description	Schema
conflicts *optional*	The device that is part of the conflict.	<ConflictDeviceObject> array

CoreSwitchDetailsInfoView

Name	Description	Schema
Description *optional*	The user-defined description.	string
Type *optional*	The type of switch.	integer (int32)
additionalAttributes *optional*	Any additional attributes.	AdditionalCoreSwitchDetailsInfo
alternateIpAddress *optional*	The alternate IP address of the switch.	string
bayId *optional*	The bay ID of the switch.	string
chassisSerialNumber *optional*	The chassis serial number.	string
chassisServiceTag *optional*	The chassis service tag for the switch.	string
checkBeacon *optional*	The beacon status.	integer (int32)
contact *optional*	The primary contact at the customer site.	string
coreSwCustomProperties *optional*	Any user-defined custom fields (name-value pairs in JSONB format).	object
coreSwDescription *optional*	The user-defined description.	string
coreSwTags *optional*	The user-defined tags for the chassis.	string
domainName *optional*	The domain name configured in the switch.	string
ethernetMask *optional*	The Ethernet mask of the core switch IP address.	string
firmware *optional*	The firmware version.	string
frameLogEnabled *optional*	Whether framelog is enabled. 1: frameLog is enabled. 0: frameLog is disabled.	integer (int32)
frameLogSize *optional*	The number of entries in the framelog	integer (int32)

Name	Description	Schema
ipAddress *optional*	The IP address of the physical switch.	string
ipAddressPrefix *optional*	The IP address prefix. This field is required to populate the prefix of the IPv6 address. Applicable only for an IPv6 address.	integer (int32)
licenseId *optional*	The license ID of the switch.	string
location *optional*	The customer site location.	string
manufacturer *optional*	The manufacturer for the switch.	string
maxNumOfBlades *optional*	The maximum number of blades that can be present in a chassis.	integer (int32)
maxPort *optional*	The maximum number of ports supported on the switch.	integer (int32)
maxVirtualSwitches *optional*	The maximum number of logical switches. 0: A non-VF switch.	integer
model *optional*	The OEM model of the chassis.	integer (int32)
modelNumber *optional*	The model number of the switch (for example, Brocade 8000).	string
name *optional*	The physical switch name.	string
numVirtualSwitches *optional*	The number of logical switches. 0: A non-VF switch.	integer
operationalStatus *optional*	The chassis operational status.	string
partNumber *optional*	The part number of the switch.	string
partition *optional*	Whether the partitions are supported in the switch.	integer (int32)
plantOfManufacturer *optional*	The manufacturer plant for the switch.	string
reachable *optional*	Whether the switch is reachable. 1: The switch is reachable. 0: The switch is unreachable.	integer (int32)
rnidSequenceNumber *optional*	The sequence number of the switch.	string
subType *optional*	The subtype of the switch. 1: DCX+ and DCX-4S+. 0: All other subtypes.	string
supplierSerialNumber *optional*	The factory serial number.	string
switchId *optional*	The switch ID.	integer
switchSerialNumber *optional*	The serial number of the switch.	string
timeZone *optional*	The time zone of the switch.	string
truFosExpiryDate *optional*	The expiration date of the TruFOS license.	string (date-time)
truFosStatus *optional*	The status of the TruFOS license. Valid values: –2: Not collected (migrated from the previous version or the TruFOS information that is not collected yet). –1: Not applicable (the FOS version is less than 9.0, an unsupported platform for TruFOS). 0: License is not installed. 1: License is installed and has expired. 2: License is valid.	integer (int32)
typeNumber *optional*	Additional details for the switch type.	string
userIpAddress *optional*	The IP address of the physical switch.	string

Name	Description	Schema
vendor *optional*	The product vendor.	string
vendorPartNumber *optional*	The vendor SKU number.	string
vendorVersion *optional*	The vendor version.	string
wwn *optional*	The physical switch WWN.	string

CustomField

Name	Description	Schema
name *optional*	The custom field name.	string
value *optional*	The custom field value.	string

DeviceEnclosureInfoView

Name	Description	Schema
additionalAttributes *optional*	Any additional attributes.	AdditionalDeviceEnclosureInfo
applications *optional*	The application that created the device enclosure.	string
autoDeleteAllowed *optional*	Whether the enclosure can be deleted by the host and VM discovery. For fabric discovery, this value must be set to 1. For host and VM discovery, this value must be set to 0. For manual discovery, this value is set based on user input. 0: The enclosure cannot be deleted by the host and VM discovery. 1: The enclosure can be deleted by the host and VM discovery.	integer (int32)
autoEnclosureName *optional*	The auto enclosure name used as a unique key to configure the auto enclosure. The name column is editable for auto enclosures.	string
baseType *optional*	The base of the device enclosure (such as storage array or server).	string
contact *optional*	The details for the contact person.	string
customProperties *optional*	Any user-defined custom fields (name-value pairs in JSONB format).	object
department *optional*	The department that is using the device enclosure.	string
description *optional*	A description of the device enclosure.	string
firmware *optional*	The firmware that is running on the device.	string
guid *optional*	The unique ID of the device enclosure.	string
health *optional*	The health of the enclosure (for example, healthy, degraded).	string
hostName *optional*	The host name that corresponds to the device enclosure.	string
id *optional*	The database ID.	integer (int32)
ipAddress *optional*	The IP address if assigned by the user.	string
location *optional*	The physical location.	string
missing *optional*	Whether the enclosure is missing.	integer (int32)
missingTime *optional*	The time when the enclosure is found to be missing.	string (date-time)
model *optional*	The model name.	string
name *optional*	The name of the device enclosure.	string

Name	Description	Schema
serialNumber *optional*	The device serial number.	string
tags *optional*	The tags associated with the device enclosure.	string
trackChanges *optional*	Whether track changes is enabled.	integer (int32)
trusted *optional*	Whether the enclosure is trusted.	integer (int32)
type *optional*	The type of device enclosure (such as storage array, server, hybrid, or null).	string
vendor *optional*	The vendor name.	string

DevicePortInfoView

Name	Description	Schema
activeZoneCount *optional*	The number of active zones to which this port belongs.	integer (int64)
activeZones *optional*	The active zones to which this port belongs.	object
activeZonesetName *optional*	The name of the active zone set in the fabric.	string
additionalAttributes *optional*	Any additional device port information.	AdditionalDevicePortInfo
agPort *optional*	Whether the port is an AG port. Valid values are 1: The device node is an AG connected to a switch in the fabric. 0: The device node is not an AG port. The default value is 0.	integer (int32)
autoEnclosureName *optional*	The port WWN.	string
customProperties *optional*	Any user-defined custom fields (name-value pairs in JSONB format).	object
description *optional*	A description of the end device or port defined by the user.	string
deviceEnclosureGuid *optional*	The GUID of the enclosure to which this port belongs.	string
deviceEnclosureId *optional*	The ID of the enclosure to which this port belongs.	integer (int32)
deviceEnclosureName *optional*	The name of the enclosure to which this port belongs.	string
deviceNodeWwn *optional*	The device node WWN.	string
deviceSymbolicName *optional*	The device symbolic name.	string
domainId *optional*	The domain ID of the switch to which this device port is connected.	integer (int32)
edgeSwitchPortWwn *optional*	This value is the same as the Switch_Port_WWN except for devices behind the AG. For devices behind the AG, it is the AG WWN N_Port mapping. This is a null field. This value is used to determine which mapping is used by the AG.	string
enclosureRole *optional*	The enclosure role (initiator or target).	string
fabricGuid *optional*	The unique fabric ID.	string
fabricId *optional*	The fabric ID to which this port belongs.	integer (int32)
fabricName *optional*	The fabric name.	string
fdmiHostName *optional*	The FDMI host name.	string
id *optional*	The unique ID of the device port.	integer (int32)
mappedBy *optional*	Whether the device has a manual or automatic enclosure map. Valid values are: 0: Automatic enclosure map. 1: Manual enclosure map.	integer (int32)
missing *optional*	Whether the device port is missing.	integer (int32)

Name	Description	Schema
missingReason *optional*	The reason the device is missing.	string
missingTime *optional*	The time when the device port became missing.	string (date-time)
name *optional*	The name of the end device or port defined by the user.	string
npivEnabled *optional*	Whether the port is NPIV enabled.	integer (int32)
number *optional*	The port number.	integer (int32)
physicalPortWwn	The physical port WWN of this device port.	string
portId *optional*	The port FC address.	string
portNumber *optional*	The connected switch port number.	integer (int32)
portRole *optional*	The port role (initiator or target).	string
proxyDevice *optional*	One of the device ports of this device node has a translated domain. That device port is set as the proxy device, and this device node is treated as virtual by assigning a value of 1 to this field. The default value is 0.	integer (int32)
simulated *optional*	Whether the device is simulated by Flow Vision.	integer (int32)
slotNumber *optional*	The connected switch slot number.	integer (int32)
speed *optional*	The currently configured port speed.	string
speedNegotiated *optional*	The currently negotiated port speed.	integer (int32)
switchId *optional*	The connected switch ID.	integer (int32)
switchName *optional*	The connected switch name.	string
switchPortId *optional*	The connected switch port ID.	integer (int32)
switchPortName *optional*	The connected switch port WWN.	string
switchPortType *optional*	The connected switch port type (for example, F_Port).	string
switchPortWwn *optional*	The switch port WWN to which this device port is physically connected. However, if the device is connected to an AG, this field contains the switch port WWN until the AG impact is applied by the application. If the AG impact is not applied, this field continues to have the switch port WWN instead of the AG port WWN.	string
switchWwn *optional*	The connected switch WWN.	string
symblicName *optional*	The port symbolic name.	string
tags *optional*	The tags associated with the device port.	string
type *optional*	The port type, U for unknown (N for N_Port, NL for NL_Port).	string
userDefinedType *optional*	The type of device (initiator or target). Note that this overrides the corresponding type of discovered end device. If the name server identifies the device as initiator+target or unknown, you can define the correct type.	string
vendor *optional*	The vendor name.	string
wwn *optional*	The port WWN.	string
zoneAlias *optional*	The zone alias name.	string

EdgeFabric

Name	Description	Schema
edgeFCRFabricID *optional*	The FCR fabric identifier assigned by the backbone fabric for this edge fabric.	integer
edgeFID *optional*	The edge FID. If virtual fabrics is not enabled for the fabric or switch, returns –1.	integer
edgeIPAddress *optional*	The IP address of the edge connected switch.	string
edgePortWwn *optional*	The edge port WWN.	string
edgeSwitchWwn *optional*	The edge switch WWN.	string
exPortWwn *optional*	The Ex_Port WWN.	string
frontDomainId *optional*	The front domain ID.	integer
translateDomainId *optional*	A list of translate domain IDs.	<integer> array

EnclosureMapping

Use this definition to map the device port to the enclosure with the following behavior:

1. If the WWN exists in SANnav and the enclosure does not exist in SANnav, create the enclosure object in SANnav and map the WWN to the enclosure.
2. If the mapping exists in SANnav and matches the SANnav data, continue to Step 3.
3. If the mapping exists in SANnav, but does not match the input data, an error occurs. Continue to Step 4. If override is true (1), then override and continue to Step 4.
4. If the WWN and the mapping exists in SANnav, input the WWN without the enclosure name and delete the mapping (unmapping).
5. If the WWN does not exist in SANnav (for example, the device is not online yet), persist the mapping and then create the mapping once the device is online.

Name	Description	Schema
devicePortWwn *required*	The WWN of the device port.	string
deviceType *required*	The enclosure type. 1: Initiator 2: Target	integer (int32)
enclosureName *optional*	The name of the enclosure to which the device port belongs.	string
override *required*	Whether to override the default behavior. 1: true (override the behavior) 0: false (do not override the behavior)	integer (int32)

EnclosureMappingInput

Name	Description	Schema
mappings *required*	The input for mapping the enclosure.	<EnclosureMapping> array

entityType

Name	Description	Schema
entityType *required*	The inventory type. Valid values: • fabric • switch • switch-port • host • host-port • storage • storage-port • chassis	string

ErrorResponse

Name	Description	Schema
code *optional*	The error code.	string
detailedErrorMessage *optional*	A detailed error message.	string
errorMessage *optional*	The error message.	string

EventAcknowledge

You can acknowledge events if the action required for the events has been addressed by sending the event identifiers and the acknowledgement or unacknowledgement notes.

Name	Description	Schema
ackBy *optional*	The user performing the action.	string
ackNotes *optional*	Any notes for the acknowledged or unacknowledged event.	string
ackedTime *optional*	The timestamp (milliseconds elapsed, counting from 1 January 1970) on the event acknowledgement.	integer (int64)
acknowledge *optional*	Whether the event needs to be acknowledged or unacknowledged.	boolean
eventIds *optional*	The event ID that needs to be acknowledged or unacknowledged.	< string > array

EventCategory

The enum provides a set of event category labels.

Type : enum (PRODUCT_EVENT, LINK_INCIDENT_EVENT, PRODUCT_AUDIT_EVENT, PRODUCT_STATUS_EVENT, SECURITY_EVENT, USER_ACTION_EVENT, MANAGEMENT_SERVER_EVENT, CORRELATION_EVENT, ALL)

EventFilter

Name	Description	Schema
description *optional*	A description of the filter.	string
lastModifiedTime *optional*	The last time the filter was modified.	string (dateTime)

Name	Description	Schema
name *optional*	The filter name.	string
severity *optional*	The trap or syslog severity.	Severity
type *optional*	The filter type.	enum (UNKNOWN, SNMP_TRAP, SYSLOG)

EventFilterColumn

The enum provides a set of columns to filter events.

Type : enum (ACKNOWLEDGED, ACKNOWLEDGED_BY, MAINTENANCE_MODE, MESSAGE_ID, ORIGIN, SPECIAL_EVENT, USER_NAME)

EventFilterObj

Name	Description	Schema
criteria *optional*	The criteria for the JSON string.	string
description *optional*	A description of the filter.	string
filterId *optional*	A unique identifier for the filter.	string
name *optional*	The filter name.	string
subtype *optional*	The filter subtype.	string
tags *optional*	Any tags associated with the filters.	string
type *optional*	The type of filter.	string
userId *optional*	The user identifier for the filter.	string

EventFilters

Name		Schema
filter *optional*	The objects used to get the filtered data.	<EventFilterObj> array

EventInfo

Name	Description	Schema
category *optional*	The category of the events.	string
categoryKey *optional*	The category key for the events.	string
eventColumn *optional*	The column name for which criteria must be set.	string
eventColumnKey *optional*	The column key for which criteria must be set.	string
id *optional*	A unique identifier for the object.	string
value *optional*	The value that must be matched for the filter.	string
valueKey *optional*	The value key associated with the value.	string

EventProductDetails

Name	Description	Schema
name *optional*	The switch name.	string
wwn *optional*	The switch WWN.	string

EventRecipientDetails

Name	Description	Schema
criteria (*optional*)	The event filter criteria based on event attributes.	ForwardingEventsFilterQueryDetails
description (*optional*)	A description about the recipient.	string
port (*required*)	The port number for the recipient. The valid range is 1 to 65535.	integer
products (*optional*)	The virtual switch WWNs. Events from these products are forwarded.	<string> array
recipientAddress (*required*)	The recipient IP address.	string
recipientType (*required*)	The recipient type (SNMP or SYSLOG).	enum (SNMP, SYSLOG)
tags (*optional*)	The user-defined comma-separated tags.	string

EventSeverityGroupSummary

Name		Schema
counter *optional*	The number of event occurrances.	integer
severityGroup *optional*	The severity group.	string

EventsAcknowledge

Name	Description	Schema
eventIdentifiers *required*	The identifier of events that must be acknowledged.	<string> array
eventNotes *required*	The notes for the acknowledging or unacknowledging event.	string

EventsFilterQuery

Name	Description	Schema
filter *optional*	Define the include event criteria that must be used to fetch the event data.	<EventsFilterQueryDetails> array

EventsFilterQueryDetails

Name	Description	Schema
includedEvents *optional*	Includes specific events while searching.	<FilterEventDetails> array

FabricDetails

Name	Description	Schema
description *optional*	A description of the fabric defined in SANnav.	string
name *optional*	The name of the fabric.	string
principalSwitchIPAddress *optional*	The IP Address of the principal switch.	string
principalSwitchWwn *optional*	The WWN of the principal switch.	string
seedSwitchIPAddress *optional*	The IP address of the seed switch.	string
seedSwitchName *optional*	The name of the seed switch.	string
seedSwitchWwn *optional*	The WWN of the seed switch.	string

FabricInfoView

Name	Description	Schema
activeZonesetName *optional*	The name of the active zone set in the fabric.	string
additionalAttributes *optional*	Any additional fabric attributes.	AdditionalFabricInfo
contact *optional*	The contact person for this fabric.	string
customProperties *optional*	Any user-defined custom fields (name-value pairs in JSONB format).	object
description *optional*	A description of the fabric.	string
enhancedTiZoneSupport *optional*	Whether the fabric has enhanced TI zone support.	integer (int32)
fabricModel *optional*	The fabric model (such as pure FOS, pure EOS, a mixed fabric with a FOS seed switch, or a mixed fabric with an EOS seed switch).	integer (int32)
health *optional*	The health status of the fabric.	string
id *optional*	The ID of the database.	integer (int32)
interopMode *optional*	The interop mode for the switch. 0 = native; 3 = open fabric.	integer (int32)
lastFabricChanged *optional*	The time when the fabric last experienced a change.	string
location *optional*	The location information.	string
managed *optional*	Whether the fabric is actively monitored. 1 = actively monitored fabric; otherwise, it is an unmonitored fabric. The default value is 1.	integer (int32)
name *optional*	The fabric name.	string
principalSwitchWwn *optional*	The WWN of the principal switch of the fabric.	string
seedSwitchAlternateIpAddress *optional*	An alternate IP address of the seed switch.	string
seedSwitchFirmwareVersion *optional*	The firmware version of the seed switch.	string
seedSwitchIpAddress *optional*	The IP address of the seed switch.	string
seedSwitchName *optional*	The name of the seed switch.	string
seedSwitchWwn *optional*	The world-wide name of the seed switch.	string

Name	Description	Schema
status *optional*	The overall operational status of the fabric. 0 = unknown; 1 = healthy; 2 = marginal, 3 = down; 5 = reachable; 6 = unreachable; 7 = degraded link.	integer (int32)
switchCount *optional*	The number of switches in the fabric.	integer (int32)
tags *optional*	Any user-defined tags.	string
trackChanges *optional*	Whether track changes are enabled for the fabric.	integer (int32)
trackingStatus *optional*	A bitmask as an integer value that represents the missing or untrusted state of fabric members, ISLs, SAN connections, device nodes, and device ports. 1 = missing switch/ISL in fabric; 2 = untrusted switch or ISL in fabric; 4 = missing initiator or port in fabric; 8 = untrusted initiator or port in fabric; 16 = missing target or port in fabric; 32 = untrusted target or port in fabric.	integer (int32)
type *optional*	The type of fabric. 0 = legacy fabric; 1 = base fabric; 2 = logical fabric; 3 = partial fabric; 4 = Ethernet fabric. The default value is 0.	integer (int32)
virtualFabricId *optional*	The virtual fabric database identifier.	integer (int32)
zoneTransactionTimeout *optional*	The number of seconds that a ZONE_TRANSACTION can be idle.	integer (int32)

FabricMemberResponse

Name		Schema
Switches *optional*	The fabric member API response that contains the fabric member (switch) information.	<SwitchDetails> array

FabricResponse

Name		Schema
Fabrics *optional*	The fabric API response that contains the metadata information for the fabric.	<FabricDetails> array

FaultEventDetailsCriteria

Defines the criteria (such as time duration) and provides options for pagination.

Name	Description	Schema
endTime *required*	The end time of the events that occurred, in milliseconds. The maximum time duration between the start and end time should not be more than 2 hours.	integer (int64)
eventProductDetails *optional*	The virtual switch WWNs that need to be considered while fetching events.	<string> array
filters *optional*	The event filter criteria based on the event attributes.	EventsFilterQuery

Name	Description	Schema
nextPageIndex *optional*	Use for pagination when the events exceed the page limit. This attribute does not have any value for the first request. If the totalRecords are more than the resultCount, then you need to set this attribute to fetch the next set of records. It has the eventID and last occurred host time values that need to be set during the subsequent request.	string
pageSize *required*	The number of events to be included in a paged result. The default is 100 events, and the maximum is 5000 events.	integer
startIndex *required*	The start index of the results set.	integer
startTime *required*	The start time of the events that occurred. Use to specify a custom start time for events, in milliseconds.	integer (int64)

FaultEventInfo

Name	Description	Schema
ackBy *optional*	The user performing the action.	string
ackNotes *optional*	The acknowledgement notes provided by the user.	string
ackedTime *optional*	The acknowledged time in long milliseconds.	integer (int64)
acknowledged *optional*	The event acknowledgement status.	string
callhomeEvent *optional*	Whether the event is a call home event.	boolean
correlatedEventIDs *optional*	The event IDs for the individual events that are part of the correlation events.	string
description *optional*	A detailed description of the event.	string
eventCategory *optional*	The event categories (switch event, switch link incident event, switch audit event, switch status event, switch security event, SANnav audit event, and SANnav application event).	string
eventCount *optional*	The number of times the event is received between the first and last host occurrence times.	integer
eventID *optional*	A unique event ID.	string
fabricName *optional*	The name of the fabric.	string
firstOccurenceHostTime *optional*	The first time the event was received by the host that is running the application.	integer (int64)
firstOccurrenceSwitchTime *optional*	The first time the event was received by the host that is running the Network Advisor application.	integer (int64)
lastOccurrenceHostTime *optional*	The last time the event was received by the host that is running the application.	integer (int64)
messageId *optional*	The message ID of the event.	string
module *optional*	The module of the event.	string
operationalStatus *optional*	The operational status of the product associated with the event.	string
origin *optional*	The event origin: SNMP trap, syslog, application event, or other.	string
portName *optional*	The name of the port that caused the event. This field is populated only for a few application events.	string
portWwn *optional*	The WWN of the port.	string
probableCause *optional*	The probable cause of the event.	string
productAddress *optional*	The IP address of the product or switch.	string

Name	Description	Schema
productName *optional*	The name of the product.	string
productUpTime *optional*	The switch uptime.	integer (int64)
recommendedActions *optional*	The recommended action to address the event.	string
ruleName *optional*	The rule name (applicable for correlation event).	string
rulePolicy *optional*	The policy of the event rule (applicable for correlation event).	string
severity *optional*	The severity of the event: emergency, alert, critical, error, warning, or info.	string
sourceAddress *optional*	If the event type is generated by SANnav, the source address is the host address where application is running. For a multinode environment, the source name is the master node host address. If the event type received is Product, the source address is the IP address of the switch.	string
sourceName *optional*	If the event type is generated by SANnav, the source name is the host name where application is running. For a multinode environment, the source name is the master node host name. If the event type received is Product, the source is the name of the switch.	string
sourceType *optional*	The source type: core switch, virtual switch, server, storage, vCenter, initiator port, target port, fabric, switch port, or others.	string
sourceWwn *optional*	The WWN for the source of the event.	string
specialEvent *optional*	Whether the event is a special event.	boolean
trapOID *optional*	The OID of the trap.	string
userName *optional*	The user ID for application events.	string

FaultEventsCriteria

Defines the criteria (such as time duration filters) and provides options for the pagination.

Name	Description	Schema
endTime *required*	The end time for events, in milliseconds.	integer (int64)
eventProductDetails *optional*	The virtual switch WWNs that must be considered while fetching events.	< string > array
eventSource *optional*	The source of the request, which can be from the dashboard or violations views.	enum (EXTERNAL_API, EVENTS)
filters *optional*	The event filter criteria based on event attributes.	EventFilters
nextPageIndex *optional*	This field is used for pagination when the events exceed the page limit.There is no value for the first request. If the totalRecords are more than the resultCount, then you need to set this to fetch the next set of records. You also need to set the eventID and last occurred host time values during the subsequent request.	string
pageSize *required*	The number of events to be included in a paged result.The default is 100 events. The maximum is 5000.	integer
startIndex *required*	The start index of the results set.	integer
startTime *required*	The start time for events, in milliseconds.	integer (int64)

FaultEventsResponse

The Event Response includes the list of matched events, the event search criteria used to fetch events, and a list of event counts by severity group.

Name	Description	Schema
eventSeverityGroupSummaryList *optional*	A list of event counts by severity group.	<EventSeverityGroupSummary> array
events *optional*	A list of matched events.	<FaultEventInfo> array
nextPageIndex *optional*	The event identifier and last occurrence host time that must be used for subsequent requests if totalRecords is more than the resultsCount.	string
pageNumber *optional*	The page number of the current page of the results set.	integer
pageSize *optional*	The page size of the results set. The default is 100 events.	integer
resultsCount *optional*	The number of records returned as per search criteria and page size.	integer
startIndex *optional*	The start index of the results set. This value is 0 for the first request and is incremented based on the subsequent requests.	integer
totalRecords *optional*	The total number of records matching the entered criteria.	integer (int64)

FCRTopologyView

Name	Description	Schema
backboneFabricID *optional*	The backbone fabric identifier.	integer
backboneSwitch *optional*	A list of backbone switches.	<BackboneSwitch> array

FilterEventDetails

Name	Description	Schema
category *optional*	The event category (SWITCH_EVENT, SWITCH_LINK_INCIDENT_EVENT, SWITCH_AUDIT_EVENT, SWITCH_STATUS_EVENT, SWITCH_SECURITY_EVENT, SANNAV_AUDIT_EVENT, and SANNAV_APPLICATION_EVENT).	string
eventColumn *optional*	The column name for the filter criteria.	EventFilterColumn
value *optional*	A value that must be matched for the filter.	string

ForwardingCategory

The enum provides the category of the event. The category can be a SANnav Application Event or a Switch Event.

Type : enum (SANNAV_APPLICATION_EVENT, SWITCH_EVENT)

ForwardingEventFilterColumn

The enum provides a set of columns to filter events.

Type : enum (DESCRIPTION, SEVERITY, SWITCH_SEVERITY, OID)

ForwardingEventsFilterQueryDetails

Name	Description	Schema
includedEvents (*optional*)	Include the specified events during search.	ForwardingFilterEventDetails
excludedEvents (*optional*)	Exclude the specified events during search.	ForwardingFilterEventDetails

ForwardingFilterEventDetails

Name	Description	Schema
category (*optional*)	Include the specified events during search.	ForwardingCategory
eventColumn (*optional*)	Exclude the specified events during search.	ForwardingEventFilterColumn
value (*optional*)	A value that must be matched for the filter. For the SANnav Application Event's Severity, valid values include: Critical, Info, Major, or Warning. For the Switch Severity, valid values include: Alert, Critical, Debug, Emergency, Error, Info, Notice, or Warning.	string

HealthSummaryDetails

Name	Description	Schema
computationTime *optional*	The last time the health score was calculated.	string
contributors *optional*	The factors contributing to the health score (such as events, configuration, status).	object
fabricGUID *optional*	The fabric GUID.	string
fabricName *optional*	The fabric name.	string
fid *optional*	The virtual fabric ID for the switch.	integer
hostGUID *optional*	The host GUID.	string
hostIPAddress *optional*	The host IP address.	string
hostName *optional*	The host name.	string
principalSwitchWWN *optional*	The principal switch WWN.	string
score *optional*	The health score of the fabric, switch, host, or storage.	integer
seedSwitchIP *optional*	The IP address of the seed switch.	string
status *optional*	The status of the entity based on a score (such as healthy, marginal, or poor).	string
storageGUID *optional*	The storage GUID.	string
storageName *optional*	The storage name.	string
switchFid *optional*	The fabric ID of the switch.	integer
switchGUID *optional*	The unique ID of the switch.	string
switchIPAddress *optional*	The IP address of the switch.	string
switchName *optional*	The name of the switch.	string
switchWWN *optional*	The WWN of the switch.	string

InventoryChassisResponse

Name	Description	Schema
chassis *optional*	The chassis details.	<CoreSwitchDetailsInfoView> array

InventoryDetails

Name	Description	Schema
fabric *optional*	The fabric to which the switch belongs.	string
principleSwitchWwn *optional*	The principle switch WWN of the fabric.	string
searchResult *optional*	The search result.	string
seedSwitchWwn *optional*	The WWN of the switch.	string
switch *optional*	The name of the switch.	string
switchIPAddress *optional*	The IP address of the switch.	string
switchWwn *optional*	The WWN of the switch.	string
vfId *optional*	The virtual fabric ID.	string

InventoryDevicePortsResponse

Name	Description	Schema
DevicePorts *optional*	Any additional device port details.	<DevicePortInfoView> array
numOfEntitiesNotReturned *optional*	The number of entities not returned due to the payload size limitation.	integer (int32)
startIndexToUse *optional*	If there are entities not returned due to size limitation, use this as the starting index to repeat the call to retrieve the remaining entities.	integer (int32)

InventoryEnclosureMappingsResponse

Name	Description	Schema
conflicts *optional*	The mapping conflicts.	<MappingConflict> array
numberOfConflicts *optional*	The number of conflicts in the provided mapping objects.	integer (int32)

InventoryEnclosuresResponse

Name	Description	Schema
Enclosures *optional*	The device enclosure information.	<DeviceEnclosureInfoView> array
numOfEntitiesNotReturned *optional*	The number of entities not returned due to the payload size limitation.	integer (int32)
startIndexToUse *optional*	If there are entities not returned due to size limitation, use this as the starting index to repeat the call to retrieve the remaining entities.	integer (int32)

InventoryFabricsResponse

Name	Description	Schema
fabrics *optional*	The fabric details.	<FabricInfoView> array

InventoryOutputResponse

Name	Description	Schema
inventory *optional*	The inventory details.	<InventoryDetails> array
searchQuery *optional*	The search parameter.	string

InventorySwitchesResponse

Name	Description	Schema
numOfEntitiesNotReturned *optional*	The number of entities not returned due to the payload size limitation.	integer (int32)
startIndexToUse *optional*	If there are entities not returned due to size limitation, use this as the starting index to repeat the call to retrieve the remaining entities.	integer (int32)
switches *optional*	The switch information.	<SwitchInfoView> array

InventorySwitchPortsResponse

Name	Description	Schema
numOfEntitiesNotReturned *optional*	The number of entities not returned due to the payload size limitation.	integer (int32)
startIndexToUse *optional*	If there are entities not returned due to size limitation, use this as the starting index to repeat the call to retrieve the remaining entities.	integer (int32)
switchPorts *optional*	The switch port information.	<SwitchPortInfoView> array

LoginResponse

Name	Description	Schema
sessionId *required*	The session ID.	string

MappingConflict

Name	Description	Schema
DevicePortWWN *optional*	The port WWN of the device.	string
deviceType *optional*	The device type.	string
existingEnclosureName *optional*	The current name of the enclosure.	string

Name	Description	Schema
issue *optional*	The mapping conflict: 1 = The mapping exists in SANnav, but does not match the input data. 2 = The WWN is invalid. 3 = The device type is invalid. 4 = The mapping failed. 5 = The enclosure name is invalid. A valid enclosure name must meet the following requirements: • The name only uses the following characters: a to z,A to Z, 0 to 9, `~!$%^&*()_+-=[]{}|:;<>?,./'". - • A maximum of 256 characters. 6 = The mapping skips the device port when it is in an unmonitored fabric or it is not present in your AOR.	integer (int32)
providedEnclosureName *optional*	The new name for the enclosure.	string

NorthboundServer

Polymorphism : Composition

Name	Description	Schema
caPublicCertificate *optional*	The CA public certificate.	string
connectionState *optional*	The server connection state. 0: Connected. 1: Streaming. 2: Error. Minimum value: 0 Maximum value: 2	integer (int32)
connectionStateReason *optional*	The connection state reason.	string
id *required*	The northbound server ID.	integer
kafkaClusterUrl *optional*	The Kafka cluster URL.	string
name *optional*	The northbound server name.	string
schemaRegistryUrl *optional*	The schema registry URL.	string
streamDetails *optional*	The stream details.	<StreamDetail> array

NorthboundServerConfig

Name	Description	Schema
caPublicCertificate *required*	The CA public certificate.	string
kafkaClusterUrl *required*	The Kafka cluster URL.	string
name *required*	The northbound server name.	string
schemaRegistryUrl *required*	The schema registry URL.	string

RequestHealthEntity

Name	Description	Schema
inventoryItem *required*	The type of entity (such as switch, fabric, host, or storage).	enum (SWITCH, FABRIC, HOST, STORAGE)
noOfRecords *optional*	The total number of records required.	integer (int32)
startIndex *optional*	The starting index of the record.	integer (int32)

score

Name	Description	Schema
ContributorType *optional*	The type of contributor.	string
descriptionDetail *optional*	The detail description.	string
score *optional*	The score.	integer

searchScope

Name	Description	Schema
scope *optional*	The scope (switchWWN, switchName, or fabricName) on which to search.	string

searchValue

Name	Description	Schema
value *optional*	The search value.	string

Severity

The enum provides the event severity lables.

Type : enum (EMERGENCY, ALERT, CRITICAL, ERROR, WARNING, INFO)

SeverityGroup

The enum provides the names of the various severity groups. The event count summaries are grouped by these severity groups.

Type : enum (ALL, ALERT, ERROR, WARNING, INFO, UNKNOWN)

SeverityLevels

The enum provides event severity lables.

Type : enum (Emergency, Alert, Critical, Error, Warning, Info)

SortableField

The enum provides the names of the event record properties that are sortable.

Type : enum (severity, sourceName, description, sourceAddress, eventCategory, eventCount, lastOccurrenceHostTime, contributors, fabricName, firstOccurenceHostTime, messageId, operationalStatus, origin, portName, probableCause, recommendedActions, ruleName, rulePolicy, acknowledged, ackBy, specialEvent, productName, productAddress, userName, sourceWwn)

SortOrder

The enum provides the sort order choices.

Type : enum (ASC, DESC, UNSORTED)

StreamConfig

Name	Description	Schema
streamAction *required*	The stream actions. 0: Disable. 1: Enable.	integer (int32)
streamType *required*	The stream ID. 1: FCPORT. 2: ETH/GIGE PORT. 3: EXTENSION. 4: SWITCH. 5: FLOW and FLOW VIOLATION.	integer (int32)

StreamDetail

Name	Description	Schema
streamName *optional*	The stream name.	string
streamState *optional*	The stream state. 0: Disabled. 1: Enabled. 2: Error.	integer (int32)
streamType *optional*	The stream type. 1: FCPORT. 2: ETH/GIGE PORT. 3: EXTENSION. 4: SWITCH. 5: FLOW and FLOW VIOLATION.	integer (int32)

StreamMetaData

Name	Description	Schema
streamState *optional*	The stream state.	<StreamState> array
streamType *optional*	The stream type.	<StreamType> array

StreamState

Name	Description	Schema
id *optional*	The stream operation ID. Minimum value: 0. Maximum value: 2.	integer (int32)
name *optional*	The stream state name.	string

StreamType

Name	Description	Schema
id *optional*	The stream ID. 1: FCPORT. 2: ETH/GIGE PORT. 3: EXTENSION. 4: SWITCH. 5: FLOW and FLOW VIOLATION.	integer (int32)
name *optional*	The stream type name.	string

SuccessResponse

Name		Schema
code *optional*	The success response code.	integer (int32)
fields *optional*	The success response fields.	string
message *optional*	The success response message.	string

SwitchDetails

Name	Description	Schema
firmwareVersion *optional*	The firmware version.	string
ipAddress *optional*	The IP address of the switch.	string
model *optional*	The switch model. Valid values are 0: Unknown. 2: Brocade.	string
modelNumber *optional*	The switch model number (such as Brocade X6-4, Brocade X6-8, or Brocade G610). The model number is OEM-specific if a custom OEM model number is available.	string
name *optional*	The name of the switch.	string
physicalSwitchWwn *optional*	The WWN of the physical switch.	string
reachable *optional*	Whether the switch is reachable from SANnav. Valid values are 0: Not reachable. 1: Reachable.	string
role *optional*	The role of the switch in the fabric. Valid values are Primary (principal switch) and Subordinate.	string
state *optional*	The state of the switch online, offline, and so on).	string
status *optional*	The computed status of the switch based on the MAPS policies defined in the switch (Operational, Degraded, and so on).	string
switchMode *optional*	The mode in which the chassis is currently operating. Valid values are 0: Switch. 2: Access Gateway.	string
virtualSwitchWwn *optional*	The WWN of the virtual switch.	string

SwitchInfoView

Name	Description	Schema
accessGatewayOperMode *optional*	The Access Gateway mode. Valid values are 0: Switch. 1: Core AG. 2: Edge AG.	integer (int32)
additionalAttributes *optional*	Any additional switch attributes.	AdditionalSwitchInfo

Name	Description	Schema
addressingMode *optional*	The switch addressing modes to assign to port addresses. Valid values are 0: Fixed. 1: Flat or 10 bit. 2: Dynamic.	integer (int32)
alternateIpAddress *optional*	The alternate IP Address of the switch.	string
baseSwitch *optional*	Whether the switch is a base switch.	integer (int32)
bayId *optional*	The bay ID of the switch.	string
chassisSerialNumber *optional*	The chassis serial number.	string
chassisServiceTag *optional*	The chassis service tag for the switch.	string
checkBeacon *optional*	The beacon status.	integer (int32)
contact *optional*	The contact name for the switch.	string
coreSwCustomProperties *optional*	The user-defined custom fields for the core switch (name-value pairs in JSONB format).	object
coreSwDescription *optional*	The user-defined description of the chassis.	string
coreSwTags *optional*	The user-defined comma-separated tags.	string
customProperties *optional*	The user-defined custom fields (name-value pairs in JSONB format).	object
Description *optional*	A description of the virtual switch.	string
discoveredPortCount *optional*	The number of managed ports in the switch.	integer (int32)
domainId *optional*	The domain ID of the switch.	integer (int32)
domainName *optional*	The domain name of the switch	string
fabricId *optional*	The unique identifier for the fabric.	integer (int32)
fabricIdidMode *optional*	The IDID mode.	integer (int32)
fabricManaged *optional*	Whether the fabric is managed. Valid values are 1: Managed. 0: Not managed.	integer (int32)
fabricName *optional*	The name of the fabric to which the switch belongs.	string
fabricType *optional*	The type of fabric. Valid values are 0: Legacy fabric. 1: Base fabric. 2: Logical fabric. 3: Partial fabric. The default value is 0.	integer (int32)
ficonSwitch *optional*	Whether FICON is supported.	integer (int32)
firmwareVersion *optional*	The firmware version.	string
health *optional*	The health status.	string
id *optional*	The unique identifier for the switch.	integer (int32)
ipAddress *optional*	The IP address (in IPv4 or IPv6 format) of the switch.	string
location *optional*	The location of the switch.	string
macAddress *optional*	The MAC address of the switch.	string
manufacturer *optional*	The manufacturer for the switch.	string
manufacturerModel *optional*	The Brocade switch model.	string
maxPort *optional*	The maximum number of ports supported on the switch.	integer (int32)
maxZoneConfigSize *optional*	The maximum size of the zone configuration.	integer (int32)
missing *optional*	Whether the switch is missing from the fabric. Valid values are 1: Missing. 0: Not missing. The default value is 0.	integer (int32)
model *optional*	The switch model type.	string
modelNumber *optional*	The model number of the switch (for example, Brocade 8000).	string

Name	Description	Schema
monitored *optional*	Whether the switch is monitored. Valid values are 0: Unmonitored. 1: Monitored.	integer (int32)
name *optional*	The switch name.	string
operationalStatus *optional*	The switch operational status.	string
partNumber *optional*	The part number of the switch.	string
physicalOperationalStatus *optional*	The chassis operational status.	string
physicalSwitchId *optional*	The physical switch identifier.	integer (int32)
physicalSwitchName *optional*	The physical switch name.	string
physicalSwitchWwn *optional*	The WWN of the switch.	string
plantOfManufacturer *optional*	The manufacturer's plant for the switch.	string
previousState *optional*	The previous state.	string
principalSwitchWwn *optional*	The WWN of the principal switch.	string
protocol *optional*	The communication protocol used to communicate with this device. Valid values are 0: Not applicable. 1: HTTP communication. 2: HTTPS communication.	integer (int32)
reachable *optional*	Whether the switch is reachable. Valid values are 1: Reachable. 0: Not reachable.	integer (int32)
rnidTag *optional*	The FICON RNID tag value.	string
role *optional*	The role of the switch (such as primary, subordinate)	string
routingPolicy *optional*	The routing policy (Advanced Performance Tuning [APT]) configured in the switch. Valid values are –1: Not applicable. The routing policy is not applicable for FOS AG switches, NOS, and mEOS switches. 0: Unknown. For non-AG FOS switches, unknown is set as the initial value. Once asset collection is successful, the corresponding value is fetched from the switch. 1: Port-based routing. 2: Device-based routing. 3: Exchange-based routing.	integer (int32)
seedSwitchWwn *optional*	The WWN of the virtual switch used as the seed switch to discover the fabric.	string
state *optional*	The state of the switch.	string
status *optional*	The status of the switch. Valid values are 0: UNKNOWN. 1: HEALTHY. 2: MARGINAL. 3: DOWN.	string
statusReason *optional*	The reason for the status, which details the contributors to the status.	string
supplierSerialNumber *optional*	The factory serial number.	string
swLicenseId *optional*	The license ID of the switch.	string
switchId *optional*	The switch identifier for the embedded port destination.	integer (int32)
switchMode *optional*	The switch mode. Valid values are 0: Switch. 2: Access Gateway.	integer (int32)
switchSerialNumber *optional*	The serial number.	string
tags *optional*	The user-defined comma-separated tags.	string
timeZone *optional*	The time zone of the switch.	string
tlsCertExpiryDate *optional*	The TLS certificate expiration time.	integer (int64)
truFosExpiryDate *optional*	The expiration date of the TruFOS license.	string (date-time)

Name	Description	Schema
truFosStatus *optional*	The status of the TruFOS license. Valid values are –2: Not collected (migrated from the previous version or the TruFOS information is not collected yet). In SANnav, it displays as a hyphen. –1: Not Applicable (FOS version is less than 9.0; TruFOS not supported). 0: License is not installed. 1: License is installed and expired. 2: Valid license.	integer (int32)
trusted *optional*	Whether the switch is a trusted member of the fabric. Valid values are 1: Trusted. 0: Not trusted. 0 is the default.	integer (int32)
type *optional*	The switch type (sw_bd_type of the switch, switchType in CLI switchShow).	integer (int32)
typeNumber *optional*	Additional details about the switch type.	string
virtualFabricId *optional*	The fabric identifier.	integer (int32)
wwn *optional*	The WWN of the switch.	string

SwitchPortInfoView

Name	Description	Schema
activeZoneCount *optional*	The number of active zones to which this port belongs.	integer (int64)
activeZones *optional*	The active zone.	object
additionalAttributes *optional*	Any additional switch port information.	AdditionalSwitchPortInfo
blocked *optional*	Whether the port is blocked.	integer (int32)
calculatedStatus *optional*	The calculated status of the port (Healthy, Down, Marginal, and so on).	string
category *optional*	The category of the switch. Valid values are 1: FC port. 2: GIGe port.	integer (int32)
connectedDeviceType *optional*	The connected device type (such as switch or Acess Gateway).	string
connectedSwitchUserPortNumber *optional*	The connected user port number of the switch.	integer (int32)
customProperties *optional*	The user-defined custom fields (name-value pairs in JSONB format).	object
defaultAreaId *optional*	The default area ID of the F_Port trunk port.	integer (int32)
degradedPort *optional*	The port is in a degraded state. Displays N/A for ports that are not online.	string
description *optional*	A description of the port.	string
disableReason *optional*	The reason the switch port is disabled.	string
disableReasonCode *optional*	The switch port disabled reason code.	integer (int32)
edgeSwitchPortWwn *optional*	For cascade AG, the WWN of the switch port (edge switch) to which the AG is logged in. For a switch or a directly connected AG, this could be null.	string
edgeSwitchWwn *optional*	For cascade AG, the WWN of the switch to which the AG is logged in. For a switch or a directly connected AG, this could be null.	string
fabricName *optional*	The name of the fabric.	string
fenced *optional*	The reason the switch port is disabled.	integer (int32)
ficonSupported *optional*	Whether the port supports FICON.	integer (int32)

Name	Description	Schema
health *optional*	The health of the port (Unmonitored, Healthy, Offline, Error, and so on).	string
id *optional*	The database ID.	integer (int32)
ipAddress *optional*	The switch IP address.	string
lastUpdate *optional*	The last update time stored as a long value. The elapsed time from 1970, in milliseconds.	integer (int64)
licensed *optional*	Whether the port is licensed.	integer (int32)
lockedPortType *optional*	The locked port type of the port. Ports can be locked down so that they can come up only in that mode.	string
logicalPortNumber *optional*	The F_Port trunk logical port number.	integer (int32)
logicalPortWwn *optional*	The logical port WWN of the F_Port trunk group.	string
macAddress *optional*	The MAC address of the port. Applicable only for GIGe ports.	string
masterPortNumber *optional*	The trunk master port number for the trunk members. For the trunk master, it has its own port number. For nontrunk ports, it has the default value (–1).	integer (int32)
maxPortSpeed *optional*	The maximum supported speed on the switch.	integer (int32)
missing *optional*	Whether the port is missing. Valid values are 0: The port is not missing. 1: The port is missing. The default value is 0.	integer (int32)
monitored *optional*	Whether the switch is monitored. Valid values are 0: Unmonitored. 1: Monitored.	integer (int32)
name *optional*	The port name.	string
npiv *optional*	Whether NPIV mode is enabled.	integer (int32)
npivCapable *optional*	The NPIV mode capability. Valid values are 1: The port is NPIV capable. 2: The NPIV license is enabled.	integer (int32)
npivEnabled *optional*	Whether the port is NPIV enabled.	integer (int32)
occupied *optional*	Whether the port is occupied.	integer (int32)
persistentDisable *optional*	Whether the port is persistently disabled.	integer (int32)
physicalPort *optional*	Whether the port is a physical port. Valid value is 0: Virtual ports (ports created by LISLs).	integer (int32)
physicalSwitchWwn *optional*	The physical switch WWN.	string
portAdminProgState *optional*	The progress state of port administrative state changes (such as enable or disable and decommission or recommission). Valid values are 0: None. 1: Enabling. 2: Disabling. 3: Decommissioning. 4: Recommissioning.	integer (int32)
portId *optional*	The port FC address.	string
portIndex *optional*	The number used for to identify the port in zoning.	integer (int32)
portMod *optional*	The port module type. Not applicable if the port does not have a GBIC installed.	string
portNumber *optional*	The logical number of the port within the slot. In case of directors, this port number is unique within the slot.	integer (int32)
previousStatus *optional*	The previous status of the port.	string
qsfpUnitNumber *optional*	The QSFP unit number for ports that have a QSFP.	integer (int32)
remoteDevice *optional*	The name or WWN of the remote device.	string
remoteDevicePortId *optional*	The remote device port (FC or GigE) ID.	integer (int32)

Name	Description	Schema
remoteNodeWwn *optional*	The WWN of the node at the remote end of this switch port.	string
remotePort *optional*	The zone alias or the name of the remote port.	string
remotePortWwn *optional*	The WWN of the port at the remote end of this switch port.	string
remoteSwitchMonitored *optional*	Whether the remote switch is monitored.	integer (int32)
remoteSwitchPortId *optional*	The remote switch port ID.	integer (int32)
slotNumber *optional*	The slot number.	integer (int32)
speed *optional*	The currently configured port speed.	string
speedNegotiated *optional*	Whether the port speed is negotiated.	integer (int32)
speedSupported *optional*	The supported port speed as a list of comma-separated values.	string
speedType *optional*	The speed type of the port. Valid values are 1: The speed is in Mb/s. 2: The speed is in Gb/s.	integer (int32)
state *optional*	The state of the port (such as online or offline).	string
status *optional*	The status of the port (No Light, No Module, Mod_inv, Online, and so on).	string
statusMessage *optional*	Any additional port-level status details (Segmented, Speed Mismatch, Trunk Master, and so on).	string
swapped *optional*	Whether the port is swapped. Valid values are 1: The port is swapped. 0: The port is not swapped.	integer (int32)
switchName *optional*	The name of the switch.	string
tags *optional*	The tags applied on this port.	string
trunkMaster *optional*	Whether the port is the trunk master. Applicable only if the port is trunked.	integer (int32)
trunked *optional*	Whether the port is trunked.	integer (int32)
trusted *optional*	Whether the port is a trusted member of the fabric. Valid values are 1: Trusted. 0: Not trusted. The default value is 0.	integer (int32)
type *optional*	The port type. The specific mode currently enabled for the port (U_Port, F_Port, E_Port, and so on).	string
virtualFabricId *optional*	The fabric identifier.	integer (int32)
virtualSwitchWwn *optional*	The virtual switch WWN.	string
wwn *optional*	The port WWN.	string
zoneAlias *optional*	The zone alias name.	string

TagsAndIdentifier

Name	Description	Schema
entityType	The inventory type. Valid values: • fabric • switch • switch-port • host • host-port • storage • storage-port • chassis	string
identifier	The identifier.	<string> array

TagsEntity

Name	Description	Schema
entityType	The inventory type. Valid values: • fabric • switch • switch-port • host • host-port • storage • storage-port • chassis	string
tagsEntityDetailsList	The tags and identifier.	<TagsEntityDetails> array

TagsEntityDetails

Name	Description	Schema
identifier *optional*	The identifier. Valid values are: For switches, LogicalSwitchWnn. For switch ports, PortWwn. For hosts and storages, Name. For host ports and storage ports, WWN. For fabrics, PrincipalSwitchWwn. For the chassis, WWN (PhysicalSwitchWwn).	string
tags *optional*	The tags based on the entity type.	string

TagsUpdate

The entity type and a list of identifiers to get tags.

Name	Description	Schema
addTags *optional*	add tags	string
entityType	The inventory type. Valid values: • fabric • switch • switch-port • host • host-port • storage • storage-port • chassis	string
identifier *optional*	The identifier.	< string > array
ids *optional*	The IDs.	< string > array
removeAllTags *optional*	Remove all existing tags.	boolean
removeTags *optional*	Remove tags.	string

TopologyResponse

Name	Schema
topology *optional*	<FCRTopologyView> array

TrapOidDetails

Name	Description	Schema
trapName *optional*	The trap MIB alias name.	string
trapOid *optional*	The trap OID.	string

UnsubscribeForwardingRecipient

Name	Description	Schema
recipientAddress *required*	The recipient IP address.	string
port *required*	The port number for the recipient. Valid range is 1 to 65535.	integer
recipientType *required*	The recipient type (SNMP or SYSLOG).	enum (SNMP, SYSLOG)

Sample Requests and Responses

This following sections provide a few sample requests and responses for using the SANnav REST API modules.

Logging In and Out

Keep the following in mind when logging in and out of the SANnav REST API.

Logging In

To log in to SANnav, you must provide a valid user name and password through an authorization header in a `POST https://<host>/external-api/v1/login/` request. If the authentication is successful, SANnav returns the session ID in the body (for example: `sessionId": "dd903934-f4d7-4eee-b05f-a7d2f48a733c`). Subsequent SANnav REST API operations must include this session ID in the request authorization header. The client applications use this token to obtain further access to the server using the persistent connection.

Request Headers

- username = guest
- password = guest
- Content-Type = application/json

URI Structure

```
POST https://<host>/external-api/v1/login/
```

Request URI

```
POST https://10.10.10.10/external-api/v1/login/
```

Request Body

There is no request body; however, you must provide a valid user name and password (such as guest / guest) through an authorization header.

Response Data

```
{
      "sessionId": "dd903934-f4d7-4eee-b05f-a7d2f48a733c"
}
```

If authentication is successful, a session ID key (for example, "sessionId": "dd903934-f4d7-4eee-b05f-a7d2f48a733c") is returned to the client in the response body. Subsequent SANnav REST API operations must include this session ID in the request authorization header. The client applications use this token to obtain further access to the server using the persistent connection.

Logging Out

To log out from SANnav, close the session using a `POST https://<host>/external-api/v1/logout/` request. You must include the session ID in the request authorization header. The SANnav address can be in the form of either an IPv4 or IPv6 address or a host:port ID.

Request Headers

- Authorization = Session ID
- Content-Type = application/json

URI Structure

```
POST https://<host>/external-api/v1/logout/
```

Request URI

```
POST https://10.10.10.10/external-api/v1/logout/
```

Request Body

There is no request body; however, you must include the session ID key in the request authorization header.

Response Data

```
{
      "ResponseCode": "AUTHENTICATION_2012",
      "ResponseId": "authentication.logout.success",
      "ResponseMessage": "User logout success"
}
```

Discovery Module

The Discovery module is used to retrieve information about the fabrics that are managed by the SANnav server as well as the seed and principal switch data for each fabric. You can also retrieve a list of switches (members) in the fabric.

Version History

This API call was introduced in SANnav Management Portal 1.1.0.

Retrieving a List of Fabrics in Your AOR

The following sample request and response use a GET request to retrieve a list of fabrics in your area of responsibility (AOR).

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/discovery/fabrics/
```

Request URI

```
GET https://10.10.10.10/external-api/v1/discovery/fabrics/
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.

```
  {
```

```
        "Fabrics": [
            {
                "name": "DevDontDisturb",
                "description": "",
                "seedSwitchName": "Sw00abcdvs123311122rr1SS",
                "seedSwitchIPAddress": "10.102.16.23",
                "seedSwitchWwn": "10:00:50:EB:1A:FD:A5:BD",
                "principalSwitchIPAddress": "10.102.16.23",
                "principalSwitchWwn": "10:00:50:EB:1A:FD:A5:BD"
            }
        ]
    }
```


Retrieving a List of Members in the Fabric

The following sample request and response use a GET request to retrieve a list of switches (members) in the fabric.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/discovery/fabric-members/?
principalSwitchWWN={principal switch wwn}
```

Request URI

```
GET https://10.10.10.10/external-api/v1/discovery/fabric-members/?
principalSwitchWWN=10:00:00:05:1E:75:AF:00
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.


```
    {
        "Switches": [
            {
                "name": "sw0sw0sw0sw022334",
                "ipAddress": "10.102.16.24",
                "physicalSwitchWwn": "10:00:50:EB:1A:FE:00:DF",
                "firmwareVersion": "v8.2.1_bld44",
                "virtualSwitchWwn": "10:00:50:EB:1A:FE:00:C0",
                "model": "2",
                "modelNumber": "Brocade X6-8",
                "switchMode": "2",
                "role": "Unknown",
                "state": "Online",
                "status": "3"
            },
```

```
            {
                "name": "Sw00abcdvs123311122rr1SS",
                "ipAddress": "10.102.16.23",
                "physicalSwitchWwn": "10:00:50:EB:1A:FD:A5:DC",
                "firmwareVersion": "v8.2.1_bld44",
                "virtualSwitchWwn": "10:00:50:EB:1A:FD:A5:BD",
                "reachable": "0",
                "model": "2",
                "modelNumber": "Brocade X6-8",
                "switchMode": "0",
                "role": "Primary",
                "state": "Online",
                "status": "0"
            },
            {
                "name": "Sw00Sw00224",
                "ipAddress": "10.102.16.25",
                "physicalSwitchWwn": "10:00:50:EB:1A:FD:D7:BD",
                "firmwareVersion": "v8.2.1_bld29",
                "virtualSwitchWwn": "10:00:50:EB:1A:FD:D7:BD",
                "model": "2",
                "modelNumber": "Brocade X6-8",
                "switchMode": "2",
                "role": "Unknown",
                "state": "Online",
                "status": "3"
            }
        ]
    }
```


FCR Module

The FCR module is used to retrieve FCR topology information for routing topologies (edge-to-edge, backbone-to-edge, and edge-to-backbone) and the relationship to the front domain and the translate domain.

Version History

This API call was introduced in SANnav Management Portal 1.1.0.

Retrieving the FCR Topology

The following sample request and response use a GET request to retrieve the FCR topology information.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/fcr/topology/
```

Request URI

```
GET https://10.10.10.10/external-api/v1/fcr/topology/
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.


```
{
    "topology": [
        {
            "backboneFabricID": 70,
            "backboneSwitch": [
                {
                    "backboneIPAddress": "10.102.18.11",
                    "backboneSwitchWwn": "10:00:88:94:71:16:36:01",
                    "EdgeFabric": [
                        {
                            "exPortWwn": "20:2F:88:94:71:16:36:01",
                            "edgeFCRFabricID": 27,
                            "edgeIPAddress": "10.102.18.148",
                            "edgeSwitchWwn": "10:00:C4:F5:7C:16:50:E5",
                            "edgePortWwn": "20:04:C4:F5:7C:16:50:E5",
                            "frontDomainId": 160,
                            "translateDomainId": [
                                200
                            ],
                            "edgeFID": 50
                        },
                        {
                            "exPortWwn": "20:29:88:94:71:16:36:01",
                            "edgeFCRFabricID": 15,
                            "edgeIPAddress": "10.102.18.147",
                            "edgeSwitchWwn": "10:00:C4:F5:7C:16:51:A5",
                            "edgePortWwn": "20:29:C4:F5:7C:16:51:A5",
                            "frontDomainId": 160,
                            "translateDomainId": [
                                200,
                                201,
                                202,
                                203,
                                204,
                                205
                            ],
                            "edgeFID": 50
                        }
                    ]
                }
            ]
        }
    ]
}
```

Fault Module

The Fault module is used to configure event forwarding, acknowledge or unacknowledge events, and display a list of filtered events.

Version History

This API call was introduced in SANnav Management Portal 1.1.0.

Acknowledging an Event

The following sample request and response use a POST request to acknowledge an event.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
POST https://<host>/external-api/v1/fault/events/acknowledge
```

Request URI

```
POST https://10.10.10.10/external-api/v1/fault/events/acknowledge
```

Request Body

```
{
    "eventIdentifiers": [
        "c0f695a7-066d-4c55-b695-a7066dec5554"
    ],
    "eventNotes": "No action is required"
}
```

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.

```
{
    "code": 200,
    "message": "Event details updated successfully.“
}
```

Unacknowledging an Event

The following sample request and response use a POST request to unacknowledge an event.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
POST https://<host>/external-api/v1/fault/events/unacknowledge
```

Request URI

```
POST https://10.10.10.10/external-api/v1/fault/events/unacknowledge
```

Request Body

```
{
     "eventIdentifiers": [
         "c0f695a7-066d-4c55-b695-a7066dec5554"
     ],
     "eventNotes": "Need to update the Administrator and take appropriate action."
}
```

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.

```
{
    "code": 200,
    "message": "Event details updated successfully.“
}
```

Subscribing to SAN Events

The following sample request and response use a POST request to create a SANnav Management Portal event recipient.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
POST https://<host>/external-api/v2/fault/events/forwarding/subscribe
```

Request URI

```
POST https://10.10.10.10/external-api/v2/fault/events/forwarding/subscribe
```

Request Body With a Filter

```
{
"criteria": {
"excludedEvents": [
        {
        "category": "SANNAV_APPLICATION_EVENT",
        "eventColumn": "DESCRIPTION",
        "value": "User"
        }
],
"includedEvents": [
        {
        "category": "SANNAV_APPLICATION_EVENT",
        "eventColumn": "SEVERITY",
```

```
        "value": "Info"
        },
        { "category": "SWITCH_EVENT",
        "eventColumn": "SEVERITY",
        "value": "Critical"
        }
]
},
"description": "Forwarding Recipient1",
"port": 162,
"products": [
],
"recipientAddress": "X.X.X.X",
"recipientType": "SNMP",
"tags": ""
}
```

Request Body Without a Filter

```
{
        "description": "Forwarding Recipient1",
        "port": 162,
        "recipientAddress": "X.X.X.X",
        "recipientType": "SNMP",
        "tags": ""
}
```

Response Data

When the operation is successful, the response has a message body similar to the following and a “200 OK” status in the headers.

```
  {
      "code": 200,
      "message": "Forwarding Target added Successfully"
  }
```

Unsubscribing to SAN Events

The following sample request and response use a POST request to unsubscribe a SANnav Management Portal event recipient.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
POST https://<host>/external-api/v2/fault/events/forwarding/unsubscribe
```

Request URI

```
POST https://10.10.10.10/external-api/v2/fault/events/forwarding/unsubscribe
```

Request Body

```
{
    "port": 162,
    "recipientAddress": "10.10.10.52",
    "recipientType": "SNMP"
}
```

Response Data

When the operation is successful, the response has a message body similar to the following and a “200 OK” status in the headers.

```
{
    "code": 200,
    "message": "Deleted the Target Successfully"
}
```

Retrieving a List of Events Using New Filter Criteria

The following sample request and response use a POST request to retrieve a list of events based on new filter criteria.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
POST https://<host>/external-api/v2/fault/events/
```

Request URI

```
POST https://10.10.10.10/external-api/v2/fault/events/
```

Request Body

```
  {
"endTime": 1623148384000,
  "eventProductDetails": ["10:00:D8:1F:CC:18:DB:34"],
"filters": {
    "filter": [
      {
        "includedEvents": [
          {
            "category": "SWITCH_EVENT",
            "eventColumn": "MESSAGE_ID",
            "value": "FSPF-1006"
          }
        ]
      }
    ]
  },
  "nextPageIndex":null,
  "pageSize": 100,
  "startIndex": 0,
  "startTime":1623143884000
}
```

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.


```
{
"events": [
  {
    "eventID": "a0ed8f8a-3ea6-41df-ad8f-8a3ea6a1df6b",
    "severity": "Major",
    "sourceName": "SANnav_Dev_G720_83",
    "sourceAddress": "10.155.39.83",
    "lastOccurrenceHostTime": 1623147439333,
    "firstOccurenceHostTime": 1623147439333,
    "eventCount": 0,
    "origin": "SANnav Polled Event",
    "eventCategory": "Switch Event",
    "description": "FSPF Version 3 not supported, input port = 8.",
    "module": "Switch Event",
    "productName": "SANnav_Dev_G720_83",
    "productAddress": "10.155.39.83",
    "operationalStatus": "HEALTHY",
    "firstOccurrenceSwitchTime": null,
    "productUpTime": 0,
    "userName": null,
    "portName": null,
    "correlatedEventIDs": null,
    "recommendedActions": null,
    "probableCause": null,
    "messageId": "FSPF-1006",
    "sourceWwn": "10:00:D8:1F:CC:18:DB:34",
    "fabricName": "Gen-7-Flow-Fabric-1@@##$$",
    "ruleName": null,
    "rulePolicy": null,
    "acknowledged": "No",
    "ackNotes": null,
    "ackBy": null,
    "ackedTime": 0,
    "callhomeEvent": false,
    "specialEvent": false,
    "sourceType": "VIRTUAL_SWITCH",
    "trapOID": null,
    "portWwn": null
  }
],
"eventSeverityGroupSummaryList": [
  {
    "severityGroup": "MAJOR",
    "counter": 10
  },
  {
    "severityGroup": "INFO",
    "counter": null
```

```
    },
    {
      "severityGroup": "ALERT",
      "counter": null
    }
  ],
  "startIndex": 0,
  "pageSize": 100,
  "resultsCount": null,
  "pageNumber": null,
  "totalRecords": 10,
  "nextPageIndex": "955fbd2e-19e0-41e4-9fbd-2e19e061e429,1623147439331"
}
```


Inventory Search Module

The Inventory Search module is used to search the SANnav inventory database with flexible query parameters. The Inventory Search module supports wildcard characters in the search.

The Inventory Search module uses the following URI structure and input parameters to search the SANnav inventory database.

URI Structure

```
GET https://<hostname>/external-api/v1/inventory/search/?
objecttype=<object_type>&searchparam=<parameter_name>&value=<any_string>&parentid=<switch_WWN>
```

The `objecttype` parameter accepts the following values:

- Switch
- SwitchPort (Note that SwitchPort requires an additional parameter (`parentId`) to return the correct value. The `parentID` parameter is the switch WWN. The `parentId` parameter is optional.)
- Device
- DevicePort
- ZoneAlias

The `searchparam` parameter accepts the following values based on the specified `objecttype` parameter:

- Switch. Valid values include: `switchWWN` or `serialnumber` (Brocade and OEM serial numbers).
- SwitchPort. Valid values include: `switchportWWN` , `switchportname` , `portfcid` (FC address), and `macaddress` .
- Device. Valid values include: `deviceNodeWWN` .
- DevicePort. Valid values include: `deviceportWWN` .
- ZoneAlias. Valid values include: `deviceNodeWWN` , `deviceportWWN` .

Version History

This API call was introduced in SANnav Management Portal 2.0.0.

Searching for a Switch by Switch WWN

The following sample request and response use a GET request to perform a wildcard (*) search for a switch using the switch WWN.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/inventory/search/?
objecttype=switch&searchparam=switchWWN&value=<switch_WWN>
```

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/search/?objecttype=switch&searchparam=switchWWN&value=10:00*
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following and a “200 OK” status in the headers.


```
{
  "Searchquery": "10:00:*",
  "Inventory": [
    {
      "switch": "Vp58_f50",
      "searchresult": "10:00:00:01:11:E1:B1:11",
      "principleswitchwwn": "10:00:00:01:11:E1:B1:11",
      "switchipaddress": "0.0.0.0",
      "switchwwn": "10:00:00:01:11:E1:B1:11",
      "seedswitchwwn": "10:00:00:22:F2:F2:22:22",
      "vfid": "50",
      "fabric": "Untitled New Fabric 2"
    },
    {
      "switch": "Brocade6505",
      "searchresult": "10:00:00:33:F3:3A:33:EF",
      "principleswitchwwn": "10:00:00:33:F3:3A:33:EF",
      "switchipaddress": "10.10.10.29",
      "switchwwn": "10:00:00:33:F3:3A:33:EF",
      "seedswitchwwn": "10:00:00:33:F3:3A:33:EF",
      "vfid": "-1",
      "fabric": "MAPS"
    },
    {
      "switch": "Ap68_f50",
      "searchresult": "10:00:00:22:F2:F2:22:22",
      "principleswitchwwn": "10:00:00:04:44:E4:B4:44",
      "switchipaddress": "10.10.10.68",
      "switchwwn": "10:00:00:22:F2:F2:22:22",
      "seedswitchwwn": "10:00:00:22:F2:F2:22:22",
      "vfid": "50",
      "fabric": "Untitled New Fabric 2"
    },
  ]
```

```
}
```

Searching for a Switch by Serial Number

The following sample request and response use a GET request to search for a switch using the serial number.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/inventory/search/?
objecttype=Switch&searchparam=serialnumber&value=<serial_number>
```

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/search/?
objecttype=Switch&searchparam=serialnumber&value=ANN0609F01K
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.

```
{
    "searchQuery": "ANN0609F01K",
    "inventory": [
        {
            "switch": "BASE_FID3_155",
            "searchResult": "ANN0609F01K",
            "principleSwitchWwn": "10:00:00:05:1E:B7:85:01",
            "switchIPAddress": "10.124.71.155",
            "switchWwn": "10:00:00:05:1E:E3:24:01",
            "seedSwitchWwn": "10:00:C4:F5:7C:B9:48:2E",
            "vfId": "3",
            "fabric": "ST_BASE"
        }
    ]
}
```

Searching for a Switch Port by the WWN of the Switch Port

The following sample request and response use a GET request to search for a switch port using the WWN of the switch port.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/inventory/search/?
objecttype=SwitchPort&searchparam=SwitchPortWWN&value=<switch_port_WWN>
```

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/search/?
objecttype=SwitchPort&searchparam=SwitchPortWWN&value=20:15:50:EB:1A:D0:50:00
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.

```
{
    "searchQuery": "20:15:50:EB:1A:D0:50:00",
    "inventory": [
        {
            "switch": "Allegiance_128",
            "searchResult": "20:15:50:EB:1A:D0:50:00",
            "principleSwitchWwn": "",
            "switchIPAddress": "",
            "switchWwn": "10:00:50:EB:1A:D0:50:FF",
            "seedSwitchWwn": "",
            "vfId": "128",
            "fabric": "new"
        }
    ]
}
```

Searching for a Switch Port by Switch Port Name

The following sample request and response use a GET request to search for a switch port using the name of the switch port.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/inventory/search/?
objecttype=SwitchPort&searchparam=SwitchPortWWN&value=<switch_port_name>
```

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/search/?
objecttype=SwitchPort&searchparam=SwitchPort&searchparam=SwitchPortName&value=slot4 port5
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.


```
{
    "searchQuery": "slot4 port5",
    "inventory": [
        {
            "switch": "Allegiance_128",
            "searchResult": "slot4 port5",
            "principleSwitchWwn": "",
            "switchIPAddress": "",
            "switchWwn": "10:00:50:EB:1A:D0:50:FF",
            "seedSwitchWwn": "",
            "vfId": "128",
            "fabric": "new"
        }
    ]
}
```


Searching for a Switch by Device Node WWN

The following sample request and response use a GET request to search for a switch using the device node WWN.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)

URI Structure

```
GET https://<host>/external-api/v1/inventory/search/?
objecttype=Device&searchparam=DeviceNODEWWN&value=<device_node_WWN>
```

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/search/?
objecttype=Device&searchparam=DeviceNodeWWN&value=20:05:00:11:0D:5B:01:00
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.


```
{
    "searchQuery": "20:05:00:11:0D:5B:01:00",
    "inventory": [
        {
            "switch": "Allegiance_128",
            "searchResult": "20:05:00:11:0D:5B:01:00",
            "principleSwitchWwn": "10:00:50:EB:1A:D0:50:00",
            "switchIPAddress": "11.11.11.11",
            "switchWwn": "10:00:50:EB:1A:D0:50:00",
            "seedSwitchWwn": "10:00:50:EB:1A:D0:50:00",
            "vfId": "128",
```

```
            "fabric": "new"
        }
    ]
}
```

Searching for a Switch by Device Port WWN

The following sample request and response use a GET request to search for a switch using the device port WWN.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/inventory/search/?
objecttype=DevicePort&searchparam=DevicePortWWN&value=<device_port_WWN>
```

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/search/?
objecttype=DevicePort&searchparam=DevicePortWWN&value=20:05:00:11:0D:5B:01:08
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.

```
{
    "searchQuery": "20:05:00:11:0D:5B:01:08",
    "inventory": [
        {
            "switch": "Allegiance_128",
            "searchResult": "20:05:00:11:0D:5B:01:08",
            "principleSwitchWwn": "10:00:50:EB:1A:D0:50:00",
            "switchIPAddress": "11.11.11.11",
            "switchWwn": "10:00:50:EB:1A:D0:50:00",
            "seedSwitchWwn": "10:00:50:EB:1A:D0:50:00",
            "vfId": "128",
            "fabric": "new"
        }
    ]
}
```

Searching for a Zone Alias by Device Node WWN

The following sample request and response use a GET request to search for a zone alias using a device node WWN in the zone alias.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/inventory/search/?
objecttype=ZoneAlias&searchparam=DeviceNODEWWN&value=<device_node_WWN>
```

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/search/?
objecttype=ZoneAlias&searchparam=DeviceNodeWWN&value=20:03:00:11:0D:BC:5F:00
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.

```
{
    "searchQuery": "20:03:00:11:0D:BC:5F:00",
    "inventory": [
        {
            "switch": "SW6510",
            "searchResult": "test_target",
            "principleSwitchWwn": "10:00:00:05:33:8C:0D:DC",
            "switchIPAddress": "11.11.11.11",
            "switchWwn": "10:00:50:EB:1A:36:F0:07",
            "seedSwitchWwn": "10:00:00:05:33:8C:0D:DC",
            "vfId": "128",
            "fabric": "Fab81*"
        }
    ]
}
```

Searching for a Zone Alias by Device Port WWN

The following sample request and response use a GET request to search for a zone alias using a device port WWN in the zone alias.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/inventory/search/?
objecttype=ZoneAlias&searchparam=DevicePortWWN&value=<device_port_WWN>
```

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/search/?
objecttype=ZoneAlias&searchparam=DevicePortWWN&value=20:03:00:11:0D:BC:5F:00
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.

```
{
    "searchQuery": "20:03:00:11:0D:BC:5F:00",
    "inventory": [
        { "switch": "SW6510",
          "searchResult": "test_target",
          "principleSwitchWwn": "10:00:00:05:33:8C:0D:DC",
          "switchIPAddress": "11.11.11.11",
          "switchWwn": "10:00:50:EB:1A:36:F0:07",
          "seedSwitchWwn": "10:00:00:05:33:8C:0D:DC",
          "vfId": "128",
          "fabric": "Fab81*"
        }
    ]
}
```

Displaying the Inventory Detail for All Enclosures

The following sample request and response use a GET request to retrieve the inventory details for all enclosures based on the search criteria.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/inventory/enclosures/?basicOnly=0
```

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/enclosures/?basicOnly=0
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following and a “200 OK” status in the headers.

```
{
    "startIndexToUse": 0,
    "numOfEntitiesNotReturned": 0,
    "Enclosures": [
        {
```

```
            "id": 1,
            "guid": "e5f99014-bfda-43df-8c7e-2dd12d41e780",
            "name": "[23] \"SANBlaze FC Port2 Targ0\"",
            "type": "Storage Array",
            "baseType": "Storage Array",
            "trackChanges": 0,
            "trusted": 1,
            "missing": 0,
            "autoDeleteAllowed": 1,
            "autoEnclosureName": "[23] \"SANBlaze FC Port2 Targ0\"",
            "additionalAttributes": {
                "createdBy": 0,
                "lastUpdateTime": 1636612978150,
                "lastUpdateModule": 0,
                "creationTime": 1636611288816,
                "virtualization": 0,
                "managedBy": 1,
                "queueDepth": 32
            }
        },
        {
            "id": 2,
            "guid": "df8d27fb-0935-4b98-9f59-73b8becf6722",
            "name": "[23] \"SANBlaze FC Port2 Targ1\"",
            "type": "Storage Array",
            "baseType": "Storage Array",
            "trackChanges": 0,
            "trusted": 1,
            "missing": 0,
            "autoDeleteAllowed": 1,
            "autoEnclosureName": "[23] \"SANBlaze FC Port2 Targ1\"",
            "additionalAttributes": {
                "createdBy": 0,
                "lastUpdateTime": 1636612978194,
                "lastUpdateModule": 0,
                "creationTime": 1636611288816,
                "virtualization": 0,
                "managedBy": 1,
                "queueDepth": 32
            }
        }
]
}
```


Mapping a Device Port to an Enclosure

The following sample request and response use a POST request to map a device port to an enclosure based on the following behavior:

- If the device port, WWN case-insensitive comparison, exists in SANnav, however the enclosure does not exist in SANnav, create the enclosure object in SANnav and map the device port to the enclosure.
- If the mapping exists in SANnav and matches the SANnav data, skip and continue to the next option.
- If the mapping exists in SANnav, however it does not match the input data and the override is set to false (0), error and continue (default) to the next option. However, if the override is set to true (1), then override the mapping and continue to the next option.
- If the WWN and the mapping exist in SANnav, input the WWN without an enclosure name, which deletes the mapping.
- If the WWN does not exist in SANnav (for example, the device is not online yet), persist and create the mapping when the device is online.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
POST https://<host>/external-api/v1/inventory/enclosures/mapping
```

Request URI

```
POST https://10.10.10.10/external-api/v1/inventory/enclosures/mapping
```

Request Body

```
{"mappings":
 [{"devicePortWwn":"21:00:F4:E9:D4:54:AC:B8","enclosureName":"FIPSdevices","deviceType":1,"override":1}]}
```

Response Data

When the operation is successful, the response has a message body similar to the following and a “200 OK” status in the headers.

```
{
    "numberOfConflicts": 0,
    "conflicts": []
}
```

Northbound Streaming Module

Northbound streaming is used to retrieve stream metadata and a list of registered northbound servers, to enable and disable a stream, and to register or remove a northbound server.

Version History

This API call was introduced in SANnav Management Portal 2.1.0.

Retrieving Stream Metadata

The following sample request and response use a GET request to retrieve a list of steam types.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json *
- Content-Type=application/json

URI Structure

```
GET https://<host>/external-api/v1/stream
```

Request URI

```
GET https://10.10.10.10/external-api/v1/stream
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.


```
{
   "streamType": [
       {
           "id": 1,
           "name": "FC Port"
       },
       {
           "id": 2,
           "name": "Eth/GigE Port"
       },
       {
           "id": 3,
           "name": "Extension Tunnel/Circuit"
       },
       {
           "id": 4,
           "name": "Switch"
       },
       {
           "id": 5,
           "name": "Flow"
       }
   ],
   "streamState": [
       {
           "id": 0,
           "name": "Disable"
       },
       {
           "id": 1,
           "name": "Enable"
       },
       {
           "id": 2,
           "name": "Error"
       }
   ]
}
```

Retrieving Information for All Registered Northbound Servers

The following sample request and response uses a GET request to retrieve information about all registered northbound servers.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json *
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/stream/servers
```

Request URI

```
GET https://10.10.10.10/external-api/v1/stream/servers
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.


```
[
    {
        "name": "nb_server",
        "kafkaClusterUrl": "10.24.24.25:9093",
        "schemaRegistryUrl": "http://10.24.24.25:8081",
        "caPublicCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURyakNDQXBhZ0F3SUJBZ0lKQU5oY3FZ
pBSkJnTlYKQkFZVEFsVlRNUk13RVFZRFZRUUlEQXBEWVd4cFptOXlibWxoTVJFd0R3WURWUVFIREFoVFlXNGdTbTl6WlRFUgpNQThHQT
SamIyMHhEREFLQmdOVkJBc01BMEpUVGpFak1DRUdBMVVFQXd3YVZXSjFiblIxCk1sOHpNbDh4TnpBdVluSnZZMkZrWlM1amIyMHdIaGN
TnpVeldoY05NakV3TWpJMk1qRXoKTnpVeldqQjdNUXN3Q1FZRFZRUUdFd0pWVXpFVE1CRUdBMVVFQ0F3S1EyRnNhV1p2Y201cFlURVJN
TJGdUlFcHZjMlV4RVRBUEJnTlZCQW9NQ0VKeWtZMjl0TVF3d0NnWURWUVFMREFOQ1UwNHhJekFoCkJnTlZCQU1NR2xWaWRXNTBkVEpmT
JOaFpHVXVZMjl0TUlJQklqQU5CZ2txaGtpRzl3MEIKQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBd2pWR09haG5Fei8zdWZnOUM5U25PVD
YmVXMwpHVmc1TzZONmxBaktPa2FLRjhoUytXeHBKanRQRW52TWtqNDh0Wkw1dHM4SUJrSnREbmt2N1Njd0FHanRtVk42CmtWemRsZWY1
hNa285c0lqVmxxaVpwQTc0dlQvdzhDS2NDaVNJdmRSeW45bUo4ejUrcjUKeEVRLzFNWWozMzZyNlE3dy9DMjhwMTVoMUR5RUo2ZTBoQ1
WGRpT1JzUWdOT0xySFQySgp3QjBiL0IxOWVCdUpnRXBJbzlXaTBZOVlXWjExWDhLdEszUWs3aVE2WlltSVNJaC8vMGhyaEV6NWdVbnk5
uUGlhR1hOT0JTdE1YbXE1VWgxdUEvR2dkY3FMOFZCR2crTlFJREFRQUJvelV3TXpBTUJnTlYKSFJNQkFmOEVBakFBTUE0R0ExVWREd0V
dOVkhTVUVEREFLQmdnckJnRUZCUWNEQVRBTgpCZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUFpVU1UT3Z2MnRTSkM3MHk4bEhaZVByaEtGdF
nRUN1lFNy9OaTY1ZGFiRmVLaDZwbnhRTGM4RVFzYnFFdmNueVAxbVM4dzRYNzFrU2o0Mk5NS1QwNjNnbzhmWTMKU1EyRWh2Y0paVUxId
TXdGWTRuN1YrcmFWVEMyeXk1N1NFUWJRN1U0S1JScXRIbTdwRQpsUncwelh3eHRkMG9GeTFkMEhPMjdDeDY0N3FXMDQyZ3ZSMjVrSzJu
nell3MGJyemdiClBoVE9QeWdGbmtIMnZFZHU1RTA2SkRuLzd6emdpNndHK09LL0dSVlZpZThTS3FrYTI4WXhCemlqWFZjVU5GQTgKL2t
hYWnNWUTdQbWhvWURDNEV1NEl1TzU4VHBaY09ZUT09Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0=",
        "id": 1,
        "connectionState": 0,
        "connectionStateReason": "SANnav successfully connected to Northbound Server.,
        "streamDetails": [
            {
                "streamType": 1,
                "streamName": "FC Port",
                "streamState": 0
            },
```

```
            {
                "streamType": 2,
                "streamName": "Eth/GigE Port",
                "streamState": 0
            },
            {
                "streamType": 3,
                "streamName": "Extension Tunnel/Circuit",
                "streamState": 0
            },
            {
                "streamType": 4,
                "streamName": "Switch",
                "streamState": 0
            },
            {
                "streamType": 5,
                "streamName": "Flow",
                "streamState": 0
            }
        ]
    }
]
```

Registering the Northbound Server

To register the northbound server, you must have the Northbound Streaming privilege with read-write permission.

The following sample request and response use a POST request to register the northbound server.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json *
- Content-Type = application/json

URI Structure

```
POST https://<host>/external-api/v1/stream
```

Request URI

```
POST https://10.10.10.10/external-api/v1/stream
```

Request Body

```
{
  "caPublicCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURyakND
pBSkJnTlYKQkFZVEFsVlRNUk13RVFZRFZRUUlEQXBEWVd4cFptOXlibWxoTVJFd0R3WURWUVFIREFoVFlXNGdTbTl6WlRFU
SamIyMHhEREFLQmdOVkJBc01BMEpUVGpFak1DRUdBMVVFQXd3YVZXSjFiblIxCk1sOHpNbDh4TnpBdVluSnZZMkZrWlM1am
TnpVeldoY05NakV3TWpJMk1qRXoKTnpVeldqQjdNUXN3Q1FZRFZRUUdFd0pWVXpFVE1CRUdBMVVFQ0F3S1EyRnNhV1p2Y20
TJGdUlFcHZjMlV4RVRBUEJnTlZCQW9NQ0VKeWtZMjl0TVF3d0NnWURWUVFMREFOQ1UwNHhJekFoCkJnTlZCQU1NR2xWaWRX
JOaFpHVXVZMjl0TUlJQklqQU5CZ2txaGtpRzl3MEIKQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBd2pWR09haG5Fei8zdWZnO
YmVXMwpHVmc1TzZONmxBaktPa2FLRjhoUytXeHBKanRQRW52TWtqNDh0Wkw1dHM4SUJrSnREbmt2N1Njd0FHanRtVk42Cmt
hNa285c0lqVmxxaVpwQTc0dlQvdzhDS2NDaVNJdmRSeW45bUo4ejUrcjUKeEVRLzFNWWozMzZyNlE3dy9DMjhwMTVoMUR5R
```

WGRpT1JzUWdOT0xySFQySgp3QjBiL0IxOWVCdUpnRXBJbzlXaTBZOVlXWjExWDhLdEszUWs3aVE2WlltSVNJaC8vMGhyaEV
uUGlhR1hOT0JTdE1YbXE1VWgxdUEvR2dkY3FMOFZCR2crTlFJREFRQUJvelV3TXpBTUJnTlYKSFJNQkFmOEVBakFBTUE0R0
dOVkhTVUVEREFLQmdnckJnRUZCUWNEQVRBTgpCZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUFpVU1UT3Z2MnRTSkM3MHk4bEhaZ
nRUN1lFNy9OaTY1ZGFiRmVLaDZwbnhRTGM4RVFzYnFFdmNueVAxbVM4dzRYNzFrU2o0Mk5NS1QwNjNnbzhmWTMKU1EyRWhc
TXdGWTRuN1YrcmFWVEMyeXk1N1NFUWJRN1U0S1JScXRIbTdwRQpsUncwelh3eHRkMG9GeTFkMEhPMjdDeDY0N3FXMDQyZ3Z
nell3MGJyemdiClBoVE9QeWdGbmtIMnZFZHU1RTA2SkRuLzd6emdpNndHK09LL0dSVlZpZThTS3FrYTI4WXhCemlqWFZjVU
hYWnNWUTdQbWhvWURDNEV1NEl1TzU4VHBaY09ZUT09Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0=",
"kafkaClusterUrl": "10.24.24.25:9093",
"name": "nb_server",
"schemaRegistryUrl: "http://10.24.24.25:8081"
}

Response Data

When the operation is successful, the response has a message body similar to the following and a “201 Created” status in the headers.

{
"name": "nb_server",
"kafkaClusterUrl": "10.24.24.25:9093",
"schemaRegistryUrl": "http://10.24.24.25:8081",
"caPublicCertificate":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURyakND
pBSkJnTlYKQkFZVEFsVlRNUk13RVFZRFZRUUlEQXBEWVd4cFptOXlibWxoTVJFd0R3WURWUVFIREFoVFlXNGdTbTl6WlRFU
SamIyMHhEREFLQmdOVkJBc01BMEpUVGpFak1DRUdBMVVFQXd3YVZXSjFiblIxCk1sOHpNbDh4TnpBdVluSnZZMkZrWlM1am
TnpVeldoY05NakV3TWpJMk1qRXoKTnpVeldqQjdNUXN3Q1FZRFZRUUdFd0pWVXpFVE1CRUdBMVVFQ0F3S1EyRnNhV1p2Y20
TJGdUlFcHZjMlV4RVRBUEJnTlZCQW9NQ0VKeWtZMjl0TVF3d0NnWURWUVFMREFOQ1UwNHhJekFoCkJnTlZCQU1NR2xWaWRX
JOaFpHVXVZMjl0TUlJQklqQU5CZ2txaGtpRzl3MEIKQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBd2pWR09haG5Fei8zdWZnO
YmVXMwpHVmc1TzZONmxBaktPa2FLRjhoUytXeHBKanRQRW52TWtqNDh0Wkw1dHM4SUJrSnREbmt2N1Njd0FHanRtVk42Cmt
hNa285c0lqVmxxaVpwQTc0dlQvdzhDS2NDaVNJdmRSeW45bUo4ejUrcjUKeEVRLzFNWWozMzZyNlE3dy9DMjhwMTVoMUR5R
WGRpT1JzUWdOT0xySFQySgp3QjBiL0IxOWVCdUpnRXBJbzlXaTBZOVlXWjExWDhLdEszUWs3aVE2WlltSVNJaC8vMGhyaEV
uUGlhR1hOT0JTdE1YbXE1VWgxdUEvR2dkY3FMOFZCR2crTlFJREFRQUJvelV3TXpBTUJnTlYKSFJNQkFmOEVBakFBTUE0R0
dOVkhTVUVEREFLQmdnckJnRUZCUWNEQVRBTgpCZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUFpVU1UT3Z2MnRTSkM3MHk4bEhaZ
nRUN1lFNy9OaTY1ZGFiRmVLaDZwbnhRTGM4RVFzYnFFdmNueVAxbVM4dzRYNzFrU2o0Mk5NS1QwNjNnbzhmWTMKU1EyRWhc
TXdGWTRuN1YrcmFWVEMyeXk1N1NFUWJRN1U0S1JScXRIbTdwRQpsUncwelh3eHRkMG9GeTFkMEhPMjdDeDY0N3FXMDQyZ3Z
nell3MGJyemdiClBoVE9QeWdGbmtIMnZFZHU1RTA2SkRuLzd6emdpNndHK09LL0dSVlZpZThTS3FrYTI4WXhCemlqWFZjVU
hYWnNWUTdQbWhvWURDNEV1NEl1TzU4VHBaY09ZUT09Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0=",
"id": 1,
"connectionState": 0,
"connectionStateReason": "SANnav successfully connected to Northbound Server.",
"streamDetails": [
{
"streamType": 1,
"streamName": "FC Port",
"streamState": 0
},
{
"streamType": 2,
"streamName": "Eth/GigE Port",
"streamState": 0
},
{
"streamType": 3,
"streamName": "Extension Tunnel/Circuit",
"streamState": 0

```
        },
        {
            "streamType": 4,
            "streamName": "Switch",
            "streamState": 0
        },
        {
            "streamType": 5,
            "streamName": "Flow",
            "streamState": 0
        }
    ]
}
```

Retrieving the Registered Northbound Server

The following sample request and response use a GET request to retrieve information about a registered northbound server.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json *
- Content-Type = application/json

URI Structure

```
GET https://<host>/external-api/v1/stream/servers/<Server_ID>
```

Request URI

```
GET https://10.10.10.10/external-api/v1/stream/servers/1
```

Request Body

There is no request body.

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.

```
    {
        "name": "nb_server",
        "kafkaClusterUrl": "10.24.24.25:9093",
        "schemaRegistryUrl": "http://10.24.24.25:8081",
        "caPublicCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURyakND
pBSkJnTlYKQkFZVEFsVlRNUk13RVFZRFZRUUlEQXBEWVd4cFptOXlibWxoTVJFd0R3WURWUVFIREFoVFlXNGdTbTl6WlRFU
SamIyMHhEREFLQmdOVkJBc01BMEpUVGpFak1DRUdBMVVFQXd3YVZXSjFiblIxCk1sOHpNbDh4TnpBdVluSnZZMkZrWlM1am
TnpVeldoY05NakV3TWpJMk1qRXoKTnpVeldqQjdNUXN3Q1FZRFZRUUdFd0pWVXpFVE1CRUdBMVVFQ0F3S1EyRnNhV1p2Y20
TJGdUlFcHZjMlV4RVRBUEJnTlZCQW9NQ0VKeWtZMjl0TVF3d0NnWURWUVFMREFOQ1UwNHhJekFoCkJnTlZCQU1NR2xWaWRX
JOaFpHVXVZMjl0TUlJQklqQU5CZ2txaGtpRzl3MEIKQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBd2pWR09haG5Fei8zdWZnO
YmVXMwpHVmc1TzZONmxBaktPa2FLRjhoUytXeHBKanRQRW52TWtqNDh0Wkw1dHM4SUJrSnREbmt2N1Njd0FHanRtVk42Cmt
hNa285c0lqVmxxaVpwQTc0dlQvdzhDS2NDaVNJdmRSeW45bUo4ejUrcjUKeEVRLzFNWWozMzZyNlE3dy9DMjhwMTVoMUR5R
WGRpT1JzUWdOT0xySFQySgp3QjBiL0IxOWVCdUpnRXBJbzlXaTBZOVlXWjExWDhLdEszUWs3aVE2WlltSVNJaC8vMGhyaEV
uUGlhR1hOT0JTdE1YbXE1VWgxdUEvR2dkY3FMOFZCR2crTlFJREFRQUJvelV3TXpBTUJnTlYKSFJNQkFmOEVBakFBTUE0R0
dOVkhTVUVEREFLQmdnckJnRUZCUWNEQVRBTgpCZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUFpVU1UT3Z2MnRTSkM3MHk4bEhaZ
```

```
nRUN1lFNy9OaTY1ZGFiRmVLaDZwbnhRTGM4RVFzYnFFdmNueVAxbVM4dzRYNzFrU2o0Mk5NS1QwNjNnbzhmWTMKU1EyRWhc
TXdGWTRuN1YrcmFWVEMyeXk1N1NFUWJRN1U0S1JScXRIbTdwRQpsUncwelh3eHRkMG9GeTFkMEhPMjdDeDY0N3FXMDQyZ3Z
nell3MGJyemdiClBoVE9QeWdGbmtIMnZFZHU1RTA2SkRuLzd6emdpNndHK09LL0dSVlZpZThTS3FrYTI4WXhCemlqWFZjVU
hYWnNWUTdQbWhvWURDNEV1NEl1TzU4VHBaY09ZUT09Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0=",
        "id": 1,
        "connectionState": 0,
        "connectionStateReason": "SANnav successfully connected to Northbound Server.,
        "streamDetails": [
            {
                "streamType": 1,
                "streamName": "FC Port",
                "streamState": 0
            },
            {
                "streamType": 2,
                "streamName": "Eth/GigE Port",
                "streamState": 0
            },
            {
                "streamType": 3,
                "streamName": "Extension Tunnel/Circuit",
                "streamState": 0
            },
            {
                "streamType": 4,
                "streamName": "Switch",
                "streamState": 0
            },
            {
                "streamType": 5,
                "streamName": "Flow",
                "streamState": 0
            }
        ]
    }
```


Enabling or Disabling a Stream

To enable or disable a stream, you must have the Northbound Streaming privilege with read-write permission.

The following sample request and response use a PUT request to enable or disable a stream.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json *
- Content-Type = application/json

URI Structure

```
PUT https://<host>/external-api/v1/stream/servers/<Server_ID>/streams
```

Request URI

```
PUT https://10.10.10.10/external-api/v1/stream/servers/1/streams
```

Request Body

To enable the stream:

```
{
 "streamType": 2,
 "streamAction": 1
}
```

To disable the stream:

```
{
 "streamType": 2,
 "streamAction": 0
}
```

Response Data

When the operation is successful, the status code “200 OK” displays in the body header.

Removing a Northbound Server

To remove the northbound server, you must have the Northbound Streaming privilege with read-write permission.

The following sample request and response use a DELETE request to remove the northbound server.

Request Headers

- Authorization = Session ID (authorization key from the login API response data) or session-less base64-encoded credentials (`Basic <base64_encoded_credentials>`)
- Accept = application/json *
- Content-Type = application/json

URI Structure

```
DELETE https://<host>/external-api/v1/stream/servers/<Server_ID>
```

Request URI

```
DELETE https://10.10.10.10/external-api/v1/stream/servers/1
```

Request Body

There is no request body.

Response Data

When the operation is successful, the status code “200 OK” displays in the body header.

Health Summary

The Health Summary module is used to retrieve the health summary for an entity (such as switch, fabric, host, or storage).

Version History

This API call was introduced in SANnav Management Portal 2.1.0.

Retrieving the Health Summary for the Requested Entity

The following sample request and response use a POST request to retrieve the health summary for the requested entity.

Request Headers

- Authorization = Session ID (authorization key from the login API response data)
- Accept = application/json
- Content-Type = application/json

URI Structure

```
POST https://<host>/external-api/v1/healthsummary/healthsummarydetails/
```

Request URI

```
POST https://10.10.10.10/external-api/v1/healthsummary/healthsummarydetails/
```

Request Body

```
    {
         "inventoryItem": "FABRIC"
     }
```

Response Data

When the operation is successful, the response has a message body similar to the following, and a “200 OK” status in the headers.


```
[
    {
        "fabricName": "Automation_1663068809180",
        "principalSwitchWWN": "10:00:D1:1F:CC:11:12:10",
        "seedSwitchIP": "10.10.10.150",
        "fid": 128,
        "fabricGUID": "11ea2231-a57e-4513-25e4-12d1f2251f73",
        "score": 88,
        "computationTime": "Tue Sep 13 11:45:09 MDT 2022",
        "contributors": [
            {
                "contributorType": "Config",
                "score": 18,
                "descriptionDetail": "[{\"recommendedAction\":\"Use Zone Management to change the access level
 to 'No Access'. If the settings are correct for your deployment, you can ignore this rule by un-selecting it
 from settings.\",\"reason\":\"Zone policy (All Access) is enabled.\",\"category\":\"Best Practices\"}]"
            },
            {
                "contributorType": "Events",
                "score": 50,
                "descriptionDetail": "[]"
            },
            {
                "contributorType": "Status",
                "score": 20,
                "descriptionDetail": "[{\"recommendedAction\":\"Check the switch health summary for reasons
 and recommended actions.\",\"reason\":\"One or more switches are in degraded health.\",\"category\":\"Status
\"}]"
            }
        ]
```

```
    },
    {
        "fabricName": "ss_G620_10_124_72_168_128",
        "principalSwitchWWN": "10:00:00:22:F2:C2:22:2C",
        "seedSwitchIP": "10.10.10.120",
        "fid": 128,
        "fabricGUID": "43db4446-5e94-4241-9a21-b04debead2a6",
        "score": 70,
        "computationTime": "Tue Sep 13 11:45:09 MDT 2022",
        "contributors": [
            {
                "contributorType": "Config",
                "score": 20,
                "descriptionDetail": "[]"
            },
            {
                "contributorType": "Events",
                "score": 50,
                "descriptionDetail": "[]"
            },
            {
                "contributorType": "Status",
                "score": 0,
                "descriptionDetail": "[{\"recommendedAction\":\"Check the switch health summary for reasons
and recommended actions.\",\"reason\":\"One or more switches are in poor health.\",\"category\":\"Status\"}]"
            }
        ]
    },
    {
        "fabricName": "Untitled New Fabric 1",
        "principalSwitchWWN": "10:00:22:14:31:AA:F1:A4",
        "seedSwitchIP": "10.10.10.120",
        "fid": 90,
        "fabricGUID": "51c06371-efb5-4001-2476-1d123e4217ca",
        "score": 70,
        "computationTime": "Tue Sep 13 11:45:09 MDT 2022",
        "contributors": [
            {
                "contributorType": "Config",
                "score": 20,
                "descriptionDetail": "[]"
            },
            {
                "contributorType": "Events",
                "score": 50,
                "descriptionDetail": "[]"
            },
            {
                "contributorType": "Status",
                "score": 0,
                "descriptionDetail": "[{\"recommendedAction\":\"Check the switch health summary for reasons
and recommended actions.\",\"reason\":\"One or more switches are in poor health.\",\"category\":\"Status\"}]"
            }
```

```
        ]
    },
    {
        "fabricName": "Base_73_120FID7",
        "principalSwitchWWN": "10:00:22:14:71:AA:F1:A5",
        "seedSwitchIP": "10.10.10.120",
        "fid": 7,
        "fabricGUID": "ca1f2247-ccd6-472a-17f0-12cdb6cfaaf4",
        "score": 70,
        "computationTime": "Tue Sep 13 11:45:09 MDT 2022",
        "contributors": [
            {
                "contributorType": "Config",
                "score": 20,
                "descriptionDetail": "[]"
            },
            {
                "contributorType": "Events",
                "score": 50,
                "descriptionDetail": "[]"
            },
            {
                "contributorType": "Status",
                "score": 0,
                "descriptionDetail": "[{\"recommendedAction\":\"Check the switch health summary for reasons
and recommended actions.\",\"reason\":\"One or more switches are in poor health.\",\"category\":\"Status\"}]"
            }
        ]
    }
]
```

Python Examples

This section provides a few Python script examples for using the SANnav REST API modules. You can execute the Python scripts, using the following command, from your Linux terminal.

```
python <python_scriptname.py> <SANnav_MP_ Server_IPAddress> <SANnav_User_Name><SANnav_password>
```

Retrieving a List of Events

Use the following Python script example to retrieve a list of events from the last 2 hours.

```
import requests
import json
import time
import base64


def __filterpayload(includedEvents=[]):
    filter = []
    filter_json = {}
    if len(includedEvents) > 0:
        filter_json['includedEvents'] = includedEvents
        filter.append(filter_json)
    payload = {'filter': filter}
    return payload


def eventpayload(nextPageIndex=None, startIndex=None, pageSize=None, startTime=None, endTime=None,
                 includedEvents=[], eventProductDetails=[]):
    payload = {'filters': __filterpayload(includedEvents),
               'startIndex': startIndex,
               'pageSize': pageSize,
               'startTime': startTime,
               'endTime': endTime,
               'nextPageIndex': nextPageIndex,
               'eventProductDetails': eventProductDetails
               }
    print(payload)
    return json.dumps(payload)


def getevents(sannav_ip=None, header=None, startTime=None, endTime=None, pageSize=500,
              startIndex=0, nextPageIndex=None, includedEvents=[]):
    payload = eventpayload(startTime=startTime, endTime=endTime, pageSize=pageSize, startIndex=startIndex,
                           nextPageIndex=nextPageIndex, includedEvents=includedEvents)
    events = requests.post(url='https://{0}/external-api/v2/fault/events/'.format(sannav_ip), data=payload,
                           headers=header, verify=False)

    assert events.status_code == 200, 'Failed to fetch the events with error code: {0}/{1}'.format(
        events, events.content)
```

```
    all_events = json.loads(events.content.decode())
    return all_events


if __name__ == '__main__':

    sannav_ip = '10.124.72.41'
    username = 'administrator'
    password = 'password'

    endtime = int(time.time())
    starttime = endtime - 7200

    ####Formating sessionless id####
    encoded_string = '{0}:{1}'.format(username, password).encode('ascii')
    encoded_id = base64.b64encode(encoded_string)
    session_id = encoded_id.decode('ascii')

    header = {'Authorization': 'Basic {0}'.format(session_id), 'Content-type': 'application/json'}

    #####Fetching the SANnav audit events####
    try:
        includedEvents = [{"category": "SANNAV_AUDIT_EVENT", "eventColumn": "ACKNOWLEDGED", "value": "No"}]
        audit_events = getevents(sannav_ip=sannav_ip, header=header, includedEvents=includedEvents,
                                 startTime='{0}000'.format(starttime), endTime='{0}000'.format(endtime))
        print('SANnav Audit Events: {0}'.format(audit_events))
    except Exception as e:
        print('Exception :{0}'.format(e))
```

Retrieving a List of Fabrics

Use the following Python script example to retrieve a list of fabrics in your AOR.

```
import requests
import json
import argparse


def login(username, password, sannav_ip):
    login_api_headers = {'Content-type': 'application/json', 'username':                  username,
 'password': password}
    resp = requests.post(url='https://{0}/external-    api/v1/login/'.format(sannav_ip),
 headers=login_api_headers,
                         verify=False)

    assert resp.status_code == 200, 'Session ID failed to generate with     error code:
 {0}'.format(resp.status_code)
    session_id = json.loads(resp.content.decode())
    return session_id


def logout(session_id, sannav_ip, header):
    logout = requests.post(url='https://{0}/external-    api/v1/logout/'.format(sannav_ip), headers=header,
```

```
                            verify=False)
    assert logout.status_code == 200, 'Failed to logout with error code:     {0}'.format(resp.status_code)


def getfabrics(sannav_ip, session_id, header):
    header = {'Authorization': session_id['sessionId'], 'Content-type':     'application/json'}
    fabric_list = requests.get(url='https://{0}/external-    api/v1/discovery/fabrics/'.format(sannav_ip),
 headers=header,
                               verify=False)

    assert fabric_list.status_code == 200, 'Failed to fetch the fabric with error code: {0}'.format(
        fabric_list.status_code)
    all_fabrics = json.loads(fabric_list.content.decode())
    return all_fabrics


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Fetchin the SANnav     details')
    parser.add_argument('IP', metavar='ip', type=str, help='SANnav IP')
    parser.add_argument('User', metavar='username', type=str,     help='SANnav username')
    parser.add_argument('Password', metavar='password', type=str,     help='SANnav password')
    args = parser.parse_args()

    sannav_ip = args.IP
    username = args.User
    password = args.Password

    try:
        ###Fetching the session id####
        session_id = login(username=username, password=password,             sannav_ip=sannav_ip)

        header = {'Authorization': session_id['sessionId'], 'Content-        type': 'application/json'}

        #####Fetching the discovered fabric list####
        all_fabrics = getfabrics(sannav_ip=sannav_ip,                     session_id=session_id,
 header=header)
        print('Fabric list: {0}'.format(all_fabrics['Fabrics']))
    except Exception as e:
        print('Failed with exception: {0}'.format(e))

    finally:
        ######Logging out the SANnav######
        print('Logoging out the session')
        logout(session_id=session_id, sannav_ip=sannav_ip, header=header)
```

Revision History

Lists the changes in the document since the initial release.

SANnav-23x-REST-API-RM100; April 28, 2023

Initial document version.

Documentation Legal Notice

This notice provides copyright and trademark information as well as legal disclaimers.

Broadcom reserves the right to make changes without further notice to any products or data herein to improve reliability, function, or design. Information furnished by Broadcom is believed to be accurate and reliable. However, Broadcom does not assume any liability arising out of the application or use of this information, nor the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.

The product described by this document may contain open source software covered by the GNU General Public License or other open source license agreements. To find out which open source software is included in Brocade products or to view the licensing terms applicable to the open source software, please download the open source attribution disclosure document in the Broadcom Support Portal. If you do not have a support account or are unable to log in, please contact your support provider for this information.


BROADCOM®