Brocade® SANnav™ Management Portal REST API and Northbound Streaming Reference Manual, 2.1.1x

Reference Manual
18 December 2020

Table of Contents

1. Overview 1
   1.1. Overall Strategy for the SANnav Management Portal REST API 1
   1.2. Using the SANnav Management Portal REST API 1
      1.2.1. Before You Begin 2
      1.2.2. Session Authorization Key 2
2. Northbound Streaming 3
   2.1. Stream Types 4
   2.2. Platform Requirements for the Northbound Server 4
   2.3. Supported Topics 4
   2.4. Stream Keys and Values 5
   2.5. Schema 6
   2.6. Migrating from SANnav 2.1.0x to 2.1.1 6
3. Resources 8
   3.1. Login And Logout 8
      3.1.1. Login 8
      3.1.2. Logout 8
   3.2. Discovery 9
      3.2.1. Listing Fabrics 9
      3.2.2. Listing Fabric Members 10
   3.3. FCR 11
      3.3.1. FCR Topology Information 11
   3.4. Fault 11
      3.4.1. Listing Events 11
      3.4.2. Listing Events (With a Newer Version of Filters) 12
      3.4.3. Subscribing Events 13
      3.4.4. Unsubscribing Events 14
      3.4.5. Bulk Update Events-acknowledgement 14
      3.4.6. Bulk Update Events-unacknowledgement 15
   3.5. Inventory 16
      3.5.1. Listing Inventory Details 16
   3.6. Stream 17
      3.6.1. Get Stream Metadata 17
      3.6.2. Get All Registered Northbound Servers 18
      3.6.3. Create a Northbound Server 19

3.6.4. Get Northbound Server Details . . . 19
3.6.5. Delete the Northbound Server . . . 20
3.6.6. Enable or Disable a Stream . . . 21
3.7. Health Summary . . . 22
3.7.1. Health Summary for Fabric, Switches, Hosts, and Storage . . . 22
4. Definitions . . . 24
4.1. BackboneSwitch . . . 24
4.2. EdgeFabric . . . 24
4.3. ErrorResponse . . . 24
4.4. EventCategory . . . 25
4.5. EventFilterColumn . . . 25
4.6. EventFilterCriteria . . . 25
4.7. EventRecipientDetails . . . 25
4.8. EventSeverityGroupSummary . . . 26
4.9. EventsAcknowledge . . . 26
4.10. EventsFilter . . . 26
4.11. EventsFilterQuery . . . 26
4.12. EventsFilterQueryDetails . . . 27
4.13. FCRTopologyView . . . 27
4.14. FabricDetails . . . 27
4.15. FabricMemberResponse . . . 27
4.16. FabricResponse . . . 28
4.17. FaultEventDetailsCriteria . . . 28
4.18. FaultEventInfo . . . 28
4.19. FaultEventsCriteria . . . 30
4.20. FaultEventsResponse . . . 30
4.21. FilterCategoryCriteria . . . 31
4.22. FilterEventDetails . . . 31
4.23. HealthSummaryDetails . . . 32
4.24. InventoryDetails . . . 32
4.25. InventoryOutputResponse . . . 33
4.26. LoginResponse . . . 33
4.27. NorthboundServer . . . 33
4.28. NorthboundServerConfig . . . 34
4.29. RequestHealthEntity . . . 34

4.30. SeverityLevels . . . 34
4.31. StreamConfig . . . 34
4.32. StreamDetail . . . 35
4.33. StreamMetaData . . . 35
4.34. StreamState . . . 35
4.35. StreamType . . . 35
4.36. SuccessResponse . . . 36
4.37. SwitchDetails . . . 36
4.38. TopologyResponse . . . 36
5. Sample Requests and Responses . . . 37
5.1. Logging In and Out . . . 37
5.1.1. Logging In . . . 37
5.1.2. Logging Out . . . 38
5.2. Discovery Module . . . 38
5.2.1. Version History . . . 38
5.2.2. Retrieving a List of Fabrics in Your AOR . . . 39
5.2.3. Retrieving a List of Members in the Fabric . . . 39
5.3. FCR Module . . . 41
5.3.1. Version History . . . 42
5.3.2. Retrieving the FCR Topology . . . 42
5.4. Fault Module . . . 43
5.4.1. Version History . . . 44
5.4.2. Acknowledging an Event . . . 44
5.4.3. Unacknowledging an Event . . . 44
5.4.4. Subscribing to SAN Events . . . 45
5.4.5. Unsubscribing to SAN Events . . . 46
5.4.6. Retrieving a Filtered List of Events . . . 47
5.4.7. Retrieving a List of Events Using New Filter Criteria . . . 49
5.5. Inventory Search Module . . . 52
5.5.1. Version History . . . 53
5.5.2. Searching for a Switch by Switch WWN . . . 53
5.5.3. Searching for a Switch by Serial Number . . . 54
5.5.4. Searching for a Switch Port by the WWN of the Switch Port . . . 55
5.5.5. Searching for a Switch Port by Switch Port Name . . . 56
5.5.6. Searching for a Switch by Device Node WWN . . . 57
5.5.7. Searching for a Switch by Device Port WWN . . . 58

5.5.8. Searching for a Zone Alias by Device Node WWN . . . 59
5.5.9. Searching for a Zone Alias by Device Port WWN . . . 60
5.6. Northbound Streaming . . . 61
5.6.1. Version History . . . 61
5.6.2. Get Stream Metadata . . . 61
5.6.3. Get All Registered Northbound Servers . . . 62
5.6.4. Register the Northbound Server . . . 64
5.6.5. Get the Registered Northbound Server . . . 66
5.6.6. Enable or Disable a Stream . . . 68
5.6.7. Remove the Northbound Server . . . 69
5.7. Python Examples . . . 70
5.7.1. Retrieving a List of Fabrics . . . 70
5.7.2. Retrieving a List of Events . . . 71
6. Revision History . . . 74
6.1. SANnav-211x-REST-API-RM100; 18 December 2020 . . . 74

Chapter 1. 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 use the REST API to build your own SANnav clients. You can also use third-party REST API clients to interact with the SANnav Management Portal.

SANnav Global View does not expose any external REST APIs.

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

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

1.1. 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 the seed and principal switch data for each fabric.
- 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.
- Stream FC and flow data (via northbound streaming).
- Display a health summary for a fabric, switch, host, or storage.

1.2. Using the SANnav Management Portal REST API

1.2.1. Before You Begin

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

To use the remaining 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.

1.2.2. 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.Please refer section Logging In for more details.

Chapter 2. Northbound Streaming

Northbound streaming provides support to securely stream performance and flow metrics from the switch 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 metrics received from the switch for PM data and raw flow metrics received from switch streaming will be streamed to the 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 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.
- Historic 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.
- SNMP-based AMP violations are not streamed to the northbound server.

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 received by SANnav during the inactive period is discarded. The heartbeat check can 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.

2.1. Stream Types

The following is a list of 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: AMP, Gen 6, and Gen 7 switch flows, AMP I/O violation metrics)

Performance statistics data for the FCPORT, ETH/GIGE PORT, EXTENSION, and SWITCH stream types are streamed to the northbound server with 5-minute granularity. For FLOW stream types, AMP flows are streamed at 10-second, 5-minute, and 6-hour granularity. AMP I/O violations, Gen 6, and Gen 7 switch flows are streamed at 10-second and 5-minute granularity.

2.2. Platform Requirements for the Northbound Server

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

- Kafka version 2.2.1
- Confluent Platform for Schema Registry and Zookeeper version 5.2.2

2.3. 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
FLOW	AMP Flow Statistics	flowmon_metrics_amp_flow	10 seconds, 5 minutes, 6 hours	16
	AMP IO Violation Statistics	flowmon_metrics_amp_flow_iov	10 seconds, 5 minutes	16
	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	
FCPORT	FC Port Statistics	perfmon_metrics_fcport	5 minutes	16
EXTENSION	Extension Tunnel Statistics	perfmon_metrics_extn_tunnel	5 minutes	16
	Extension Circuit Statistics	perfmon_metrics_extn_circuit	5 minutes	16

Stream Type	Stream Description	Topic Name	Stream Interval	Recommended Partition Count
ETH/GIGE	TE and GigE Port Statistics	perfmon_metrics_ethport	5 minutes	16
SWITCH	Switch Statistics	perfmon_metrics_switch	5 minutes	16

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.

2.4. 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.

Topic	Kafka Message Key	Kafka Message Value
flowmon_metrics_amp_flow	SID + DID + LUN (Same key received from switch streaming) Example: 3b0000_030200_1	amp_flow_statistics
flowmon_metrics_amp_flow_iov	SID + DID + LUN (Same key received from switch streaming) Example: 3b0000_030200_1	amp_io_violation_statistics
flowmon_metrics_switch_flow	SID + DID + LUN (Same key received from switch streaming) Example: 3b0000_030200_1	fibrechannel_flow_statistics
perfmon_metrics_fcport	Port WWN Example: 20:14:50:EB:1A:0C:C8:4F	fibrechannel_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_ethport	MAC Address Example: 00:27:F8:68:D1:90	ethernet_port_statistics
perfmon_metrics_switch	Switch WWN Example: 10:00:00:05:33:e7:cf:8f	fibrechannel_switch_statistics

2.5. Schema

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

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

- AMP flow schema (`amp_flow_external_schema.avsc`)
- AMP IO violations schema (`amp_flow_io_violations_external_schema.avsc`)
- 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`)
- Switch schema (`switch_external_schema.avsc`)
- TE and GigE port schema (`eth_port_external_schema.avsc`)

2.6. Migrating from SANnav 2.1.0x to 2.1.1

Because of topic name and schema changes, if you want to migrate from SANnav 2.1.0x to SANnav 2.1.1, you must perform the following steps:

1. Remove the northbound server registration from SANnav.
2. Remove the following northbound topics from the northbound Kafka cluster:
    - ¬ perfmon.metrics.fcport
    - ¬ perfmon.metrics.extn.tunnel
    - ¬ perfmon.metrics.extn.circuit
    - ¬ perfmon.metrics.ethport

- perfmon.metrics.switch
- flowmon.metrics.amp.flow
- flowmon.metrics.amp.flow,iov
- flowmon.metrics.switch.flow
- heartbeat.metrics.switch

3. Remove all 2.1.0x schemas from the northbound schema registry.
4. Create the northbound server registration in the SANnav 2.1.1.

Chapter 3. Resources

3.1. Login and Logout

3.1.1. Login

```
POST /external-api/v1/login/
```

Description

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

Parameters

Type	Name	Description	Schema
Header	**username** *required*	User name to log in using external-api.	string
Header	**password** *required*	Password to log in using external-api.	string (password)

Responses

HTTP Code	Description	Schema
200	Return the session ID for successful login	LoginResponse
401	Unauthorized	No Content
500	Unexpected error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

3.1.2. Logout

```
POST /external-api/v1/logout/
```

Description

Logs out of the SANnav Management Portal server.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string

Responses

HTTP Code	Description	Schema
200	Successfully logged out	No Content
401	Unauthorized	ErrorResponse
500	Unexpected error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

3.2. Discovery

3.2.1. Listing Fabrics

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

Description

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

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string

Responses

HTTP Code	Description	Schema
200	All fabrics details are retrieved successfully	FabricResponse
401	Unauthorized	No Content
500	Unexpected error occurred	ErrorResponse

Produces

- `application/json`

3.2.2. Listing Fabric Members

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

Description

Gets all the fabric members in the fabric.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string
Query	**principalSwitchWWN** *required*	Principal switch WWN	string

Responses

HTTP Code	Description	Schema
200	All switch details are retrieved successfully	FabricMemberResponse
400	Bad request	ErrorResponse
401	Unauthorized	No Content
404	Not found	ErrorResponse
500	Unexpected error occurred	ErrorResponse

Produces

- `application/json`

3.3. FCR

3.3.1. FCR Topology Information

```
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*	Session ID	string

Responses

HTTP Code	Description	Schema
200	FCR topology information.	TopologyResponse
401	Unauthorized	No Content
500	Unexpected error occurred	ErrorResponse

Produces

- `application/json`

3.4. Fault

3.4.1. Listing Events

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

Description

Lists the events matching the given criteria.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string
Body	**eventCriteria** *optional*	The attributes to be considered for fetching events.	FaultEventsCriteria

Responses

HTTP Code	Description	Schema
200	Events Response Object. It contains a list of event counts for each severity group	FaultEventsResponse
400	Bad request	ErrorResponse
401	Unauthorized	No Content
500	Unexpected error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

3.4.2. Listing Events (With a Newer Version of Filters)

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

Description

Lists the events matching the given criteria.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string
Body	**eventCriteria** *optional*	The attributes to be considered for fetching events.	FaultEventDetailsCriteria

Responses

HTTP Code	Description	Schema
200	Events Response Object. It contains a list of event counts for each severity group	FaultEventsResponse
400	Bad Request	ErrorResponse
401	Unauthorized	No Content
500	Unexpected error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

3.4.3. Subscribing Events

```
POST /external-api/v1/fault/events/forwarding/subscribe
```

Description

Create the trap/syslog forwarding targets.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string
Body	**messageDetail** *required*	Event forwarding target details object.	EventRecipientDetails

Responses

HTTP Code	Description	Schema
200	Successful creation of resource	SuccessResponse
400	Bad request	ErrorResponse
401	Unauthorized	No Content
412	Precondition failed; will occur if recipient already exists.	ErrorResponse
500	Unexpected error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

3.4.4. Unsubscribing Events

```
POST /external-api/v1/fault/events/forwarding/unsubscribe
```

Description

Deletes the trap/syslog forwarding targets.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string
Body	**messageDetail** *required*	Event forwarding target details object.	EventRecipientDetails

Responses

HTTP Code	Description	Schema
200	Successful deletion of resource	SuccessResponse
400	Bad request	ErrorResponse
401	Unauthorized	No Content
412	Precondition failed, will occur if recipient does not exist.	ErrorResponse
500	Unexpected error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

3.4.5. Bulk Update Events-acknowledgement

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

Description

Updates the given acknowledgement details for a given list of events.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string
Body	**acknowledgeEvents** *required*	Payload having event identifiers and acknowledgment/unacknowledge notes.	EventsAcknowledge

Responses

HTTP Code	Description	Schema
200	Successfully updated acknowledgement details in events.	SuccessResponse
400	Bad request	ErrorResponse
401	Unauthorized	No Content
500	Unexpected error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

3.4.6. Bulk Update Events-unacknowledgement

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

Description

Updates the given unacknowledgement details for a given list of events.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string

Type	Name	Description	Schema
Body	**unacknowledgeEvents** *required*	Payload having event identifiers and acknowledgment/unacknowledge notes.	EventsAcknowledge

Responses

HTTP Code	Description	Schema
200	Successfully updated unacknowledgement details in events.	SuccessResponse
400	Bad request	ErrorResponse
401	Unauthorized	No Content
500	Unexpected error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

3.5. Inventory

3.5.1. Listing Inventory Details

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

Description

Gets all the details of the object type searched.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string
Query	**objecttype** *required*	Context in which the search should be applied	enum (Switch, SwitchPort, Device, DevicePort, ZoneAlias)

Type	Name	Description	Schema
Query	**searchparam** *required*	Search parameter	enum (switchWWN, serialnumber, switchportWWN, switchportname, portfcid, macaddress, deviceNodeWWN, deviceportWWN)
Query	**value** *required*	Value	string
Query	**parentId** *optional*	ParentId refers to switch WWN. It is the parameter for switch port and it is optional.	string

Responses

HTTP Code	Description	Schema
200	Inventory details are retrieved successfully	InventoryOutputResponse
400	Bad Request	ErrorResponse
401	Unauthorized	No Content
404	Not Found	ErrorResponse
500	Unexpected error occurred	ErrorResponse

Produces

- `application/json`

3.6. Stream

3.6.1. Get Stream Metadata

```
GET /external-api/v1/stream
```

Description

API to get 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*	Session ID	string

Responses

HTTP Code	Description	Schema
200	Ok. The requested StreamMetaData in the response body.	StreamMetaData
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

Produces

- `application/json`

3.6.2. Get All Registered Northbound Servers

```
GET /external-api/v1/stream/servers
```

Description

API to get 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*	Session ID	string

Responses

HTTP Code	Description	Schema
200	Ok. The requested list of northbound server data is in the response body.	< 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

Produces

- `application/json`

3.6.3. Create a Northbound Server

```
POST /external-api/v1/stream/servers
```

Description

API to create a northbound server.
The request should contain the northbound server name, Kafka Broker URL, Schema Registry URL, and encoded CA Public Certificate in Base64 format. Returns northbound server details received in the request with the server ID and list of stream details.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string
Body	**northboundServerConfig** *required*	Northbound server configuration	NorthboundServerConfig

Responses

HTTP Code	Description	Schema
201	Created. 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

Consumes

- `application/json`

Produces

- `application/json`

3.6.4. Get Northbound Server Details

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

Description

API to get northbound server details.

Returns 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*	Session ID	string
Path	**serverId** *required*	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 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

Produces

- `application/json`

3.6.5. Delete the Northbound Server

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

Description

API to delete the northbound server.

All active streaming will be stopped for the requested server, and the northbound server will be removed.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string

Type	Name	Description	Schema
Path	**serverId** *required*	Server Identifier	integer

Responses

HTTP Code	Description	Schema
200	Ok. The requested northbound server is removed from the system.	No Content
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

Produces

- `application/json`

3.6.6. Enable or Disable a Stream

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

Description

API to enable or disable a stream.

Enabling a stream will start streaming to the requested northbound server.

Disabling a stream will stop streaming to the requested northbound server.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string
Path	**serverId** *required*	Server Identifier	integer
Body	**streamConfig** *required*	Stream configuration	StreamConfig

Responses

HTTP Code	Description	Schema
200	Ok. Stream state updated successfully.	No Content

HTTP Code	Description	Schema
400	Bad Request. The requested stream is already in that state. The response body contains additional information.	ErrorResponse
401	Unauthorized. The Authorization request parameter is not valid. The response body contains additional information.	ErrorResponse
404	Not Found. Invalid Request Parameter. 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

Consumes

- `application/json`

Produces

- `application/json`

3.7. Health Summary

3.7.1. Health Summary for Fabric, Switches, Hosts, and Storage

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

Description

Gets a health summary for the requested entity.

Parameters

Type	Name	Description	Schema
Header	**Authorization** *required*	Session ID	string
Body	**requestEntity** *required*	Health Summary Details	RequestHealthEntity

Responses

HTTP Code	Description	Schema
200	Health Summary are retrieved successfully	HealthSummaryDetails
400	Bad Request	ErrorResponse
401	Unauthorized	No Content

HTTP Code	Description	Schema
404	Not Found	ErrorResponse
500	Unexpected error occurred	ErrorResponse

Consumes

- `application/json`

Produces

- `application/json`

Chapter 4. Definitions

4.1. BackboneSwitch

Backbone switch details.

Name	Description	Schema
backboneIPAddress *optional*	Router IP address	string
backboneSwitchWwn *optional*	Backbone switch WWN	string
edgeFabric *optional*	List of edge fabrics.	< EdgeFabric > array

4.2. EdgeFabric

Edge fabric details.

Name	Description	Schema
exPortWwn *optional*	Ex_Port WWN	string
edgeFCRFabricID *optional*	FCR fabric ID assigned by the backbone fabric for this edge fabric	integer
edgeIPAddress *optional*	Edge connected switch IP address	string
edgeSwitchWwn *optional*	Edge switch WWN	string
edgePortWwn *optional*	Edge port WWN	string
frontDomainId *optional*	Front domain ID	integer
translateDomainId *optional*	List of translate domain IDs	< integer > array
edgeFID *optional*	Edge FID (This will return -1 if virtual fabrics is not enabled for the fabric/switch)	integer

4.3. ErrorResponse

Name	Description	Schema
code *optional*	Error code	string
errorMessage *optional*	Error message	string
detailedErrorMessage *optional*	Detailed error message	string

4.4. 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)

4.5. 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)

4.6. EventFilterCriteria

Name	Description	Schema
categories *optional*	Include specific event categories while searching.	FilterCategoryCriteria
includedEvents *optional*	Include specific events while searching.	< FilterEventDetails > array
excludedEvents *optional*	Exclude specific events while searching.	< FilterEventDetails > array

4.7. EventRecipientDetails

Attributes for event forwarding.

Name	Description	Schema
recipientAddress *required*	Recipient IP address.	string
port *required*	Port number for the recipient. Valid range is 1-65535.	integer
recipientType *required*	Either SNMP or SYSLOG	enum (SNMP, SYSLOG)
description *optional*	Description about the recipient.	string
includeSourceAddress *optional*	Adds the source address in the trap.	boolean
forwardApplicationEvents *optional*	Enables or disables forwarding of application events.	boolean
forwardCorrelatedEvents *optional*	Enables or disables forwarding of correlated events.	boolean

Name	Description	Schema
severityLevel *optional*	This parameter defines the trap or syslog severity level.The trap or syslog severity levels are:Emergency,Alert,Critical,Error,Warning, and Info. If you select Info as severity level for a forwarding destination, SANnav sends all events with all severity levels. If you select Alert as severity level, SANnav sends only Alert and Emergency severity events.	SeverityLevels
products *optional*	Virtual switch WWNs. Events from these products will be forwarded.	< string > array
eventDescriptionPattern *optional*	Checks for the regular expression of the event description that must be considered while forwarding the events.	string
trapOid *optional*	TRAP OID details. Only these traps will be forwarded.	< string > array

4.8. EventSeverityGroupSummary

Maps the number of event occurrences with their severity group.

Name	Schema
severityGroup *optional*	string
counter *optional*	integer

4.9. EventsAcknowledge

Defines the auto acknowledge details.

Name	Description	Schema
eventIdentifiers *required*	ID of events that must be acknowledged.	< string > array
eventNotes *required*	Notes for the acknowledging/unacknowledging event.	string

4.10. EventsFilter

The user can define categories/severities or include/exclude event criteria that must be used while fetching the event data.

Name	Schema
filter *optional*	< EventFilterCriteria > array

4.11. EventsFilterQuery

User can define include event criteria that needs to be used while fetching the event data.

Name	Schema
filter *optional*	< EventsFilterQueryDetails > array

4.12. EventsFilterQueryDetails

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

4.13. FCRTopologyView

FCR Topology view.

Name	Description	Schema
backboneFabricID *optional*	Backbone Fabric ID	integer
backboneSwitch *optional*	List of backbone switches.	< BackboneSwitch > array

4.14. FabricDetails

Attributes of a fabric.

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

4.15. FabricMemberResponse

The wrapper object used for fabric member API response that contains fabric member (switch) information.

Name	Schema
Switches *optional*	< SwitchDetails > array

4.16. FabricResponse

The wrapper object used for fabric API response that contains fabric's meta-data information.

Name	Schema
Fabrics *optional*	< FabricDetails > array

4.17. FaultEventDetailsCriteria

Defines criteria like time duration filters and provides options for paginated result.

Name	Description	Schema
filters *optional*	Event filter criteria based on event attributes.	EventsFilterQuery
startIndex *optional*	Indicates the start index of results set.	integer
pageSize *optional*	Indicates the number of entries to be included in a paged result. Default is 100 events and maximum of 5000.	integer
startTime *optional*	Start time of the events occurred. Use this field to specify custom start time for events in milliseconds.	integer (int64)
endTime *optional*	End time of the events occurred in milliseconds. The maximum time duration between start and end time should not be more than two hours.	integer (int64)
nextPageIndex *optional*	Used for pagination when events exceed page limit. Attribute will not have any value for first request. If the totalRecords are more than the resultCount then user will have to set this to fetch next set of records. It will have the eventID and lastoccurred host time value that needs to be set during subsequent request.	string
eventProductDetails *optional*	Virtual switch WWNs that needs to be considered while fetching events.	< string > array

4.18. FaultEventInfo

Event Details info object.

Name	Description	Schema
eventID *optional*	Unique event ID.	string
severity *optional*	Severity for the event:Emergency, Alert, Critical, Error, Warning or Info.	string
sourceName *optional*	If the event type is generated by SANnav, source name will be host name where application is running; and for multinode environment, source name will be master node host name. If the event type is received as Product, the source will be the name of the switch.	string

Name	Description	Schema
sourceAddress *optional*	If the event type is generated by SANnav, source address will be host address where application is running; and for multinode environment, source name will be master node host address. If the event type is received as Product, the source address will be the IP of the switch.	string
lastOccurrenceHostTime *optional*	Last time the event was received by the host that is running the application.	integer (int64)
firstOccurenceHostTime *optional*	First time the event was received by the host that is running the application.	integer (int64)
eventCount *optional*	Number of times the event is received between the first and last host occurrence times.	integer
origin *optional*	The event origin:SNMP Trap, Syslog, Application Event or Other.	string
eventCategory *optional*	The event category such as Product Event, Link incident Event, Product Audit Event, Product Status Event, Security Event, User Action Event, Management Server Event, and Correlation Event.	string
description *optional*	The detailed description of the event.	string
module *optional*	The module of the event.	string
productName *optional*	Name of the product.	string
productAddress *optional*	The IP address of the product/switch.	string
operationalStatus *optional*	The operational status of the product associated with the event.	string
firstOccurrenceSwitchTime *optional*	First time the event was received by the host that is running the Network Advisor application.	integer (int64)
productUpTime *optional*	Switch uptime.	integer (int64)
userName *optional*	User ID for application events.	string
portName *optional*	Name of the port that caused this event. Will be populated only for a few application events.	string
correlatedEventIDs *optional*	Captures the event IDs for the individual events that are part of correlation events.	string
recommendedActions *optional*	Recommended action to address the event.	string
probableCause *optional*	Describes the probable cause of the event.	string
messageId *optional*	Describes the message ID of the event.	string
sourceWwn *optional*	WWN for the source of the event.	string
fabricName *optional*	The name of the fabric.	string
ruleName *optional*	Rule name(Applicable for Correlation event).	string

Name	Description	Schema
rulePolicy *optional*	Policy of the event rule(Applicable for Correlation event).	string
acknowledged *optional*	Event acknowledgement status.	integer
ackNotes *optional*	Acknowledgement notes provided by the user.	string
ackBy *optional*	Details of the user performing the action.	string
ackedTime *optional*	Acknowledged time in long milliseconds.	integer (int64)
callhomeEvent *optional*	Indicates if the event is a call home event.	integer
specialEvent *optional*	Indicates if the event is a special event.	integer
sourceType *optional*	The source type:Core Switch, Virtual Switch, Server, Storage, Vcenter, Initiator port, Target port, Fabric, Switch Port, or Others.	string
trapOID *optional*	The OID of the trap	string
portWwn *optional*	The WWN of the port.	string

4.19. FaultEventsCriteria

Defines criteria like time duration filters and provides options for paginated result.

Name	Description	Schema
filters *optional*	Event filter criteria based on event attributes.	EventsFilter
startIndex *optional*	The start index of the results set.	integer
pageSize *optional*	The number of entries to be included in a paged result.The Default is 100 events, and maximum is 5000.	integer
startTime *optional*	This parameter is used to specify the start time for events in milliseconds.	integer (int64)
endTime *optional*	This parameter is used to specify the end time for events in milliseconds.	integer (int64)
nextPageIndex *optional*	Used for pagination when events exceed page limit.The attribute will not have any value for first request. If the totalRecords are more than the resultCount then user will have to set this to fetch next set of records. It will have the eventID and lastoccurred host time value that needs to be set during subsequent request.	string
eventProductDetails *optional*	Virtual switch WWNs that must be considered while fetching events.	< string > array
eventSource *optional*	source of the request , which can be from DASHBOARD view / Violations view.	enum (EXTERNAL_API, EVENTS)

4.20. 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
events *optional*	A list of matched events.	< FaultEventInfo > array
eventSeverityGroupSummaryList *optional*	A list of event counts by severity group.	< EventSeverityGroupSummary > array
startIndex *optional*	The start index of the results set. Should be 0 for the first request and can be incremented based on the subsequent requests.	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
pageNumber *optional*	The page number of the current page of the results set.	integer
totalRecords *optional*	The total number of records matching the entered criteria.	integer (int64)
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

4.21. FilterCategoryCriteria

Name	Description	Schema
correlationEvent *optional*	Selected severity for the correlation event:Emergency, Alert, Critical, Error, Warning and Info.	< SeverityLevels > array
managementServerEvent *optional*	Selected severity for the management server event.	< SeverityLevels > array
userActionEvent *optional*	Selected severity for the user action event.	< SeverityLevels > array
securityEvent *optional*	Selected severity for the security event.	< SeverityLevels > array
productStatusEvent *optional*	Selected severity for the product status event.	< SeverityLevels > array
productEvent *optional*	Selected severity for the product event.	< SeverityLevels > array
productAuditEvent *optional*	Selected severity for the product audit event.	< SeverityLevels > array
linkIncidentEvent *optional*	Selected severity for the link incident event.	< SeverityLevels > array

4.22. FilterEventDetails

Name	Description	Schema
category *optional*	Category of the events.	EventCategory
eventColumn *optional*	Column name for the filter criteria.	EventFilterColumn

Name	Description	Schema
value *optional*	A value that has to be matched for the filter.	string

4.23. HealthSummaryDetails

Name	Description	Schema
fabricName *optional*	Fabric name	string
principalSwitchWWN *optional*	Principal Switch WWN	string
seedSwitchIP *optional*	IP address of the seed switch	string
fid *optional*	Virtual fabric ID for switch	integer
fabricGUID *optional*	Fabric GUID	string
score *optional*	Denotes health score of either fabric, switch,host or storage	integer
computationTime *optional*	Latest time when health score is calculated	string
contributors *optional*	Factors contributing the health score, like Events,Config,Status etc..	object
status *optional*	Denotes status of entity based on score, can be healthy, marginal or poor	string
switchName *optional*	Switch name	string
switchIPAddress *optional*	Switch IP	string
switchWWN *optional*	WWN of the switch	string
switchFid *optional*	Fabric ID of the switch	integer
switchGUID *optional*	GUID of the switch	string
hostName *optional*	Host name	string
hostIPAddress *optional*	Host IP	string
hostGUID *optional*	Host GUID	string
storageName *optional*	Storage name	string
storageGUID *optional*	Storage GUID	string

4.24. InventoryDetails

Name	Description	Schema
searchResult *optional*	Search result	string
principleSwitchWwn *optional*	Principle switch WWN of the fabric	string
switchIPAddress *optional*	IP address of the switch	string
switchWwn *optional*	WWN of the switch	string
switch *optional*	Name of the Switch	string
seedSwitchWwn *optional*	Seed switch WWN of the Switch	string
vfId *optional*	ID of the Fabric	string
fabric *optional*	Fabric of the switch	string

4.25. InventoryOutputResponse

Name	Description	Schema
searchQuery *optional*	Search parameter	string
inventory *optional*		< InventoryDetails > array

4.26. LoginResponse

Name	Description	Schema
sessionId *optional*	Session ID	string

4.27. NorthboundServer

Polymorphism : Composition

Name	Description	Schema
name *optional*	Northbound Server Name	string
kafkaClusterUrl *optional*	Kafka Cluster URL	string
schemaRegistryUrl *optional*	Schema Registry URL	string
caPublicCertificate *optional*	CA Public Certificate	string
id *required*	Northbound Server Id	integer

Name	Description	Schema
connectionState *optional*	Server Connection State. 0 - Connected, 1 - Streaming and 2 - Error. **Minimum value** : 0 **Maximum value** : 2	integer (int32)
connectionStateReason *optional*	Connection State Reason	string
streamDetails *optional*	Stream Details	< StreamDetail > array

4.28. NorthboundServerConfig

Name	Description	Schema
name *optional*	Northbound Server Name	string
kafkaClusterUrl *optional*	Kafka Cluster URL	string
schemaRegistryUrl *optional*	Schema Registry URL	string
caPublicCertificate *optional*	CA Public Certificate	string

4.29. RequestHealthEntity

Name	Description	Schema
inventoryItem *optional*	Type of an entity SWITCH, FABRIC, HOST, STORAGE	enum (SWITCH, FABRIC, HOST, STORAGE)
noOfRecords *optional*	Total no of records required	integer (int32)
startIndex *optional*	Starting index of the record	integer (int32)

4.30. SeverityLevels

The enum provides event severity lables.

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

4.31. StreamConfig

Name	Description	Schema
streamType *optional*	Stream ID. 1 - FCPORT, 2 - ETH/GIGE PORT, 3 - EXTENSION, 4 - SWITCH and 5 - FLOW and FLOW VIOLATION. **Minimum value** : 1 **Maximum value** : 5	integer (int32)

Name	Description	Schema
streamAction *optional*	Stream Action. 0 - DISABLE and 1 - ENABLE. **Minimum value** : 0 **Maximum value** : 1	integer (int32)

4.32. StreamDetail

Name	Description	Schema
streamType *optional*	Stream Type. 1 - FCPORT, 2 - ETH/GIGE PORT, 3 - EXTENSION, 4 - SWITCH and 5 - FLOW and FLOW VIOLATION. **Minimum value** : 1 **Maximum value** : 5	integer (int32)
streamName *optional*	Stream Name	string
streamState *optional*	Stream State. 0 - DISABLED, 1 - ENABLED and 2 - ERROR. **Minimum value** : 0 **Maximum value** : 2	integer (int32)

4.33. StreamMetaData

Name	Description	Schema
streamType *optional*	Stream Type	< StreamType > array
streamState *optional*	Stream State	< StreamState > array

4.34. StreamState

Name	Description	Schema
id *optional*	Stream Operation Id. **Minimum value** : 0 **Maximum value** : 2	integer (int32)
name *optional*	Stream State	string

4.35. StreamType

Name	Description	Schema
id *optional*	Stream Id. 1 - FCPORT, 2 - ETH/GIGE PORT, 3 - EXTENSION, 4 - SWITCH, 5 - FLOW and FLOW VIOLATION. **Minimum value** : 1 **Maximum value** : 5	integer (int32)
name *optional*	Stream Name	string

4.36. SuccessResponse

Name	Schema
code *optional*	integer (int32)
message *optional*	string
fields *optional*	string

4.37. SwitchDetails

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

4.38. TopologyResponse

Name	Schema
topology *optional*	< FCRTopologyView > array

Chapter 5. Sample Requests and Responses

This section provides a few sample requests and responses for using the SANnav REST API modules.

5.1. Logging In and Out

The following items should be kept in mind when logging in and out of the SANnav REST API.

5.1.1. 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.

5.1.2. 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"
}
```

5.2. Discovery Module

The Discovery module is used to retrieve information about the fabrics 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.

5.2.1. Version History

This API call was introduced in SANnav Management Portal 1.1.0.

5.2.2. Retrieving a List of Fabrics in Your AOR

The following sample request and response uses 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)
- 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"
        }
    ]
}
```

5.2.3. Retrieving a List of Members in the Fabric

The following sample request and response uses 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)
- 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"
        }
    ]
}
```


5.3. 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 front domain and translate domain.

5.3.1. Version History

This API call was introduced in SANnav Management Portal 1.1.0.

5.3.2. Retrieving the FCR Topology

Request Headers

- Authorization = Session ID (authorization key from the login API response data)
- 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
                        }
                    ]
                }
            ]
        }
    ]
}
```


5.4. Fault Module

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

5.4.1. Version History

This API call was introduced in SANnav Management Portal 1.1.0.

5.4.2. Acknowledging an Event

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

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/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.“
}
```

5.4.3. Unacknowledging an Event

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

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/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.“
}
```

5.4.4. Subscribing to SAN Events

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

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/fault/events/forwarding/subscribe
```

Request URI

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

Request Body

```
{
    "forwardApplicationEvents": true,
    "forwardCorrelatedEvents": true,
    "includeSourceAddress": true,
    "port": 162,
    "products": [],
    "recipientAddress": "10.155.41.52",
    "recipientType": "SNMP",
    "severityLevel": "Info",
    "trapOid": []
}
```

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"
}
```

5.4.5. Unsubscribing to SAN Events

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

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/fault/events/forwarding/unsubscribe
```

Request URI

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

Request Body

```
{
    "port": 162,
    "recipientAddress": "10.155.41.52",
    "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"
}
```

5.4.6. Retrieving a Filtered List of Events

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

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/fault/events/
```

Request URI

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

Request Body

```
{
    "endTime": 1537269900000,
    "eventProductDetails": [],
    "filters": {
        "filter": [
            {
                "categories": {
                    "correlationEvent": [],
                    "linkIncidentEvent": [],
                    "managementServerEvent": [
                        "Emergency","Alert","Error","Info"
                    ],
                    "productAuditEvent": [
                        "Emergency","Alert","Error","Info"
                    ],
                    "productEvent": [
                        "Emergency","Alert","Error","Info"
                    ],
                    "productStatusEvent": [
                        "Info"
                    ],
                    "securityEvent": [
                        "Emergency","Alert","Error"
                    ],
                    "userActionEvent": [
                        "Emergency","Alert","Error","Info"
                    ]
                },
                "excludedEvents": [],
                "includedEvents": [
                    {
                        "category": "ALL",
                        "eventColumn": "ACKNOWLEDGED",
                        "value": "No"
                    }
                ]
            }
        ]
    },
    "pageSize": 100,
    "startIndex": 0,
    "startTime": 1537266600000
}
```

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": "9521d239-78ac-4a78-a1d2-3978ac8a7811",
            "severity": "Error",
            "sourceName": "rhel74_34186",
            "sourceAddress": "10.124.72.36",
            "lastOccurrenceHostTime": 1538130213556,
            "firstOccurenceHostTime": 1538130213556,
            "origin": "Application Event",
            "eventCategory": "Management Server Event",
            "description": "Failed to register SNMP(trap) for the switch
10.155.34.16. The Trap Recipient table is full.",
            "correlatedEventIDs": "",
            "recommendedActions": "",
            "probableCause": "",
            "messageId": "SSMP-EVNT-2002",
            "ruleName": "",
            "rulePolicy": "",
            "ackNotes": "",
            "ackBy": "",
            "sourceType": "OTHERS"
        }
    ],
    "eventSeverityGroupSummaryList": [
        {
            "severityGroup": "ERROR",
            "counter": 59
        },
        {
            "severityGroup": "INFO",
            "counter": 2
        },
        {
            "severityGroup": "ALERT"
        }
    ],
    "startIndex": 0,
    "pageSize": 100,
    "totalRecords": 61,
    "nextPageIndex": "3a42ff68-c588-417f-82ff-68c588617fa3,1538126737459"
}
```


5.4.7. Retrieving a List of Events Using New Filter Criteria

The following sample request and response uses 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)

- 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

```
{
    "pageSize": 100,
    "startIndex": 0,
    "endTime": 1580365960956,
    "eventProductDetails": [],
    "startTime": 1580365050956,
    "acknowledgedEvents": false,
    "filters": {
        "filter": [
            {
                "filterId": "temp-1580365761032",
                "action": "add",
                "name": "events filter 1",
                "tags": "",
                "description": "",
                "filterTypeId": 2,
                "criteria": "{\"includedEvents\":[{\"id\":\"Product EventMessage
IDMAPS-1003\",\"category\":\"Product Event\",\"categoryKey\":\"Product
Event\",\"eventColumn\":\"Message
ID\",\"eventColumnKey\":\"messageId\",\"value\":\"MAPS-
1003\",\"valueKey\":\"MAPS-1003\"}]}",
                "userId": 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.

```
{
    "events": [
        {
            "eventID" : "958b8c44-8333-4be7-8b8c-4483332be73e",
            "eventCategory" : "Product Event",
```

```
            "description" : "Chassis, Condition=CHASSIS(BAD_FAN>=0), Current
Value:[BAD_FAN,0], RuleName=C_BF_G1_N_x_RTExxx, Dashboard Category=N/A.",
            "rasLogId" : null,
            "origin" : "Other",
            "severity" : "Warning",
            "messageId" : "MAPS-1003",
            "meId" : 5,
            "firstOccurenceHostTime" : 1586855036228,
            "sourceWwn" : "10:00:00:05:1E:E3:24:01",
            "virtualFabricId" : -1,
            "trapOID" : null,
            "sourceAddress" : "10.124.71.155",
            "sourceName" : "BASE_FID3_155",
            "productUpTime" : 0,
            "checksum" : -968057585060735614,
            "portName" : null,
            "module" : "Product Event",
            "reasonCode" : -1,
            "slot" : -1,
            "port" : -1,
            "unit" : -1,
            "fruCode" : -1,
            "fruPostion" : -1,
            "eventKey" : "SWE-MAPS-1003",
            "nodeWwn" : null,
            "operationalStatus" : "Down",
            "eventNumber" : 14100860,
            "userName" : null,
            "userID" : -1,
            "productAddress" : "10.124.71.155",
            "audit" : null,
            "contributors" : null,
            "recommendedActions" : null,
            "probableCause" : null,
            "fabricName" : null,
            "fabricID" : 3,
            "eventRuleId" : 0,
            "drop" : -1,
            "lastOccurrenceHostTime" : 1586855036242,
            "eventCount" : 50,
            "callhomeEvent" : -1,
            "portWwn" : null,
            "macAddress" : null,
            "productTime" : 1586882185000,
            "sourceType" : "VIRTUAL_SWITCH",
            "sourceGuid" : "956133ec-3c55-4743-a133-ec3c554743f8",
            "fabricGuid" : "2156a9a6-b0e9-47ae-96a9-a6b0e927ae00",
            "productName" : null,
            "isMaintenanceMode" : 0,
            "acknowledged" : -1,
            "ackedTime" : -1,
            "ackNotes" : "",
            "ackBy" : "",
            "specialEvent" : -1,
            "isLogRequired" : 1,
            "isEmailSupported" : -1,
```

```
            "emailBody" : "",
            "emailSubject" : "",
            "emailRecepients" : "",
            "isSupportSaveRequired" : -1,
            "ruleName" : "",
            "rulePolicy" : "",
            "correlatedEventIDs" : ""
        }
    ],
    "eventSeverityGroupSummaryList": [
        {
            "severityGroup": "ERROR",
            "counter": null
        },
        {
            "severityGroup": "INFO",
            "counter": 1
        },
        {
            "severityGroup": "ALERT",
            "counter": null
        }
    ],
    "startIndex": 0,
    "pageSize": 100,
    "resultsCount": null,
    "pageNumber": null,
    "totalRecords": 1,
    "nextPageIndex": "fbe480a5-7582-48e5-a480-a57582b8e504,1580365776103"
}
```


5.5. 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>&parenti
d=<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.

5.5.1. Version History

This API call was introduced in SANnav Management Portal 2.0.0.

5.5.2. Searching for a Switch by Switch WWN

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

Request Headers

- Authorization = Session ID (authorization key from the login API response data)

URI Structure

```
GET https://<hostname>/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=10:00:00:10:F1:F2:11:01
```

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": "10:00:00:27:F8:F4:13:01",
     "inventory": [
         {
             "switch": "BASE_FID3_148",
             "searchResult": "10:00:00:27:F8:F4:13:01",
             "principleSwitchWwn": "10:00:00:05:1E:B7:85:01",
             "switchIPAddress": "10.124.71.148",
             "switchWwn": "10:00:00:27:F8:F4:13:01",
             "seedSwitchWwn": "10:00:C4:F5:7C:B9:48:2E",
             "vfId": "3",
             "fabric": "ST_BASE"
         }
     ]
}
```


5.5.3. Searching for a Switch by Serial Number

The following sample request and response uses 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)

URI Structure

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

Request URI

```
GET https://10.102.16.202/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"
        }
    ]
}
```


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

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

Request Headers

- Authorization = Session ID (authorization key from the login API response data)

URI Structure

```
https://<hostname>/external-api/v1/inventory/search/?
objecttype=Switch&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"
        }
    ]
}
```


5.5.5. Searching for a Switch Port by Switch Port Name

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

Request Headers

- Authorization = Session ID (authorization key from the login API response data)

URI Structure

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

Request URI

```
GET https://10.10.10.10/external-api/v1/inventory/search/?
objecttype=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"
        }
    ]
}
```


5.5.6. Searching for a Switch by Device Node WWN

The following sample request and response uses 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)

URI Structure

```
GET https://<hostname>/external-api/v1/inventory/search/?
objecttype=Switch&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"
        }
    ]
}
```


5.5.7. Searching for a Switch by Device Port WWN

The following sample request and response uses 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)

URI Structure

```
GET https://<hostname>/external-api/v1/inventory/search/?
objecttype=Switch&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"
        }
    ]
}
```

5.5.8. Searching for a Zone Alias by Device Node WWN

The following sample request and response uses 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)

URI Structure

```
GET https://<hostname>/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*"
        }
    ]
}
```

5.5.9. Searching for a Zone Alias by Device Port WWN

The following sample request and response uses 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)

URI Structure

```
GET https://<hostname>/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*"
        }
    ]
}
```

5.6. Northbound Streaming

5.6.1. Version History

This API call was introduced in SANnav Management Portal 2.1.0.

5.6.2. Get Stream Metadata

The following sample request and response uses a GET request to get steam types.

Request Headers

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

URI Structure

```
GET https://<SANnav IP>/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 body similar to the following, and a “200 OK” status in the

header.


```
{
   "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"
       }
   ]
}
```


5.6.3. Get All Registered Northbound Servers

The following sample request and response uses a GET request to get all registered northbound servers.

Request Headers

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

URI Structure

```
GET https://<SANnav IP>/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 body similar to the following, and a “200 OK” status in the header.


```
[
   {
       "name": "nb_server",
       "kafkaClusterUrl": "10.24.24.25:9093",
       "schemaRegistryUrl": "http://10.24.24.25:8081",
       "caPublicCertificate":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURyakNDQXBhZ0F3SUJBZ0lKQU5oY3FZRnluOG9
wTUEwR0NTcUdTSWIzRFFFQkN3VUFNSHN4Q3pBSkJnTlYKQkFZVEFsVlRNUk13RVFZRFZRUUlEQXBEWVd
4cFptOXlibWxoTVJFd0R3WURWUVFIREFoVFlXNGdTbTl6WlRFUgpNQThHQTFVRUNnd0lRbkp2WVdSamI
yMHhEREFLQmdOVkJBc01BMEpUVGpFak1DRUdBMVVFQXd3YVZXSjFiblIxCk1sOHpNbDh4TnpBdVluSnZ
ZMkZrWlM1amIyMHdIaGNOTWpBd01qSTNNakV6TnpVeldoY05NakV3TWpJMk1qRXoKTnpVeldqQjdNUXN
3Q1FZRFZRUUdFd0pWVXpFVE1CRUdBMVVFQ0F3S1EyRnNhV1p2Y201cFlURVJNQThHQTFVRQpCd3dJVTJ
GdUlFcHZjMlV4RVRBUEJnTlZCQW9NQ0VKeWtZMjl0TVF3d0NnWURWUVFMREFOQ1UwNHhJekFoCkJnTlZ
CQU1NR2xWaWRXNTBkVEpmTXpKZk1UY3dMbUp5YjJOaFpHVXVZMjl0TUlJQklqQU5CZ2txaGtpRzl3MEI
KQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBd2pWR09haG5Fei8zdWZnOUM5U25PVDZFZ095M2tUQ1NjRXR
1YmVXMwpHVmc1TzZONmxBaktPa2FLRjhoUytXeHBKanRQRW52TWtqNDh0Wkw1dHM4SUJrSnREbmt2N1N
jd0FHanRtVk42CmtWemRsZWY1RURFUzVwU3A4WVdOMkhNa285c0lqVmxxaVpwQTc0dlQvdzhDS2NDaVN
JdmRSeW45bUo4ejUrcjUKeEVRLzFNWWozMzZyNlE3dy9DMjhwMTVoMUR5RUo2ZTBoQ1BhRWdtYmo3TmV
zdFBRWGRpT1JzUWdOT0xySFQySgp3QjBiL0IxOWVCdUpnRXBJbzlXaTBZOVlXWjExWDhLdEszUWs3aVE
2WlltSVNJaC8vMGhyaEV6NWdVbnk5Q0R3Ck1tb1J6Wjk4K3JuUGlhR1hOT0JTdE1YbXE1VWgxdUEvR2d
kY3FMOFZCR2crTlFJREFRQUJvelV3TXpBTUJnTlYKSFJNQkFmOEVBakFBTUE0R0ExVWREd0VCL3dRRUF
3SUZvREFUQmdOVkhTVUVEREFLQmdnckJnRUZCUWNEQVRBTgpCZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUF
pVU1UT3Z2MnRTSkM3MHk4bEhaZVByaEtGdFRuR3hSNHlEUzUzQTFpCnRUN1lFNy9OaTY1ZGFiRmVLaDZ
wbnhRTGM4RVFzYnFFdmNueVAxbVM4dzRYNzFrU2o0Mk5NS1QwNjNnbzhmWTMKU1EyRWh2Y0paVUxIdEt
WODFOSVZBdnBqazFVTXdGWTRuN1YrcmFWVEMyeXk1N1NFUWJRN1U0S1JScXRIbTdwRQpsUncwelh3eHR
kMG9GeTFkMEhPMjdDeDY0N3FXMDQyZ3ZSMjVrSzJuN05SQUkwdVpLVzMxRmNnell3MGJyemdiClBoVE9
QeWdGbmtIMnZFZHU1RTA2SkRuLzd6emdpNndHK09LL0dSVlZpZThTS3FrYTI4WXhCemlqWFZjVU5GQTg
KL2tnWjNDbFl5anZtSURoc1hYWnNWUTdQbWhvWURDNEV1NEl1TzU4VHBaY09ZUT09Ci0tLS0tRU5EIEN
FUlRJRklDQVRFLS0tLS0=",
       "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
            }
        ]
    }
]
```

5.6.4. Register the Northbound Server

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

Request Headers

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

URI Structure

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

Request URI

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

Request Body


```
{

“caPublicCertificate":"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURyakNDQXBhZ0F3S
UJBZ0lKQU5oY3FZRnluOG9wTUEwR0NTcUdTSWIzRFFFQkN3VUFNSHN4Q3pBSkJnTlYKQkFZVEFsVlRNU
k13RVFZRFZRUUlEQXBEWVd4cFptOXlibWxoTVJFd0R3WURWUVFIREFoVFlXNGdTbTl6WlRFUgpNQThHQ
TFVRUNnd0lRbkp2WVdSamIyMHhEREFLQmdOVkJBc01BMEpUVGpFak1DRUdBMVVFQXd3YVZXSjFiblIxC
k1sOHpNbDh4TnpBdVluSnZZMkZrWlM1amIyMHdIaGNOTWpBd01qSTNNakV6TnpVeldoY05NakV3TWpJM
k1qRXoKTnpVeldqQjdNUXN3Q1FZRFZRUUdFd0pWVXpFVE1CRUdBMVVFQ0F3S1EyRnNhV1p2Y201cFlUR
VJNQThHQTFVRQpCd3dJVTJGdUlFcHZjMlV4RVRBUEJnTlZCQW9NQ0VKeWIyRmtZMjl0TVF3d0NnWURWU
VFMREFOQ1UwNHhJekFoCkJnTlZCQU1NR2xWaWRXNTBkVEpmTXpKZk1UY3dMbUp5YjJOaFpHVXVZMjl0T
UlJQklqQU5CZ2txaGtpRzl3MEIKQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBd2pWR09haG5Fei8zdWZnO
UM5U25PVDZFZ095M2tUQ1NjRXR1YmVXMwpHVmc1TzZONmxBaktPa2FLRjhoUytXeHBKanRQRW52TWtqN
Dh0Wkw1dHM4SUJrSnREbmt2N1Njd0FHanRtVk42CmtWemRsZWY1RURFUzVwU3A4WVdOMkhNa285c0lqV
mxxaVpwQTc0dlQvdzhDS2NDaVNJdmRSeW45bUo4ejUrcjUKeEVRLzFNWWozMzZyNlE3dy9DMjhwMTVoM
UR5RUo2ZTBoQ1BhRWdtYmo3TmVzdFBRWGRpT1JzUWdOT0xySFQySgp3QjBiL0IxOWVCdUpnRXBJbzlXa
TBZOVlXWjExWDhLdEszUWs3aVE2WlltSVNJaC8vMGhyaEV6NWdVbnk5Q0R3Ck1tb1J6Wjk4K3JuUGlhR
1hOT0JTdE1YbXE1VWgxdUEvR2dkY3FMOFZCR2crTlFJREFRQUJvelV3TXpBTUJnTlYKSFJNQkFmOEVBa
kFBTUE0R0ExVWREd0VCL3dRRUF3SUZvREFUQmdOVkhTVUVEREFLQmdnckJnRUZCUWNEQVRBTgpCZ2txa
GtpRzl3MEJBUXNGQUFPQ0FRRUFpVU1UT3Z2MnRTSkM3MHk4bEhaZVByaEtGdFRuR3hSNHlEUzUzQTFpC
nRUN1lFNy9OaTY1ZGFiRmVLaDZwbnhRTGM4RVFzYnFFdmNueVAxbVM4dzRYNzFrU2o0Mk5NS1QwNjNnb
zhmWTMKU1EyRWh2Y0paVUxIdEtWODFOSVZBdnBqazFVTXdGWTRuN1YrcmFWVEMyeXk1N1NFUWJRN1U0S
1JScXRIbTdwRQpsUncwelh3eHRkMG9GeTFkMEhPMjdDeDY0N3FXMDQyZ3ZSMjVrSzJuN05SQUkwdVpLV
zMxRmNnell3MGJyemdiClBoVE9QeWdGbmtIMnZFZHU1RTA2SkRuLzd6emdpNndHK09LL0dSVlZpZThTS
3FrYTI4WXhCemlqWFZjVU5GQTgKL2tnWjNDbFl5anZtSURoc1hYWnNWUTdQbWhvWURDNEV1NEl1TzU4V
HBaY09ZUT09Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0=",
"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 header.


```
{
   "name": "nb_server",
   "kafkaClusterUrl": "10.24.24.25:9093",
   "schemaRegistryUrl": "http://10.24.24.25:8081",
   "caPublicCertificate":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURyakNDQXBhZ0F3SUJBZ0lKQU5oY3FZRnluOG9
wTUEwR0NTcUdTSWIzRFFFQkN3VUFNSHN4Q3pBSkJnTlYKQkFZVEFsVlRNUk13RVFZRFZRUUlEQXBEWVd
4cFptOXlibWxoTVJFd0R3WURWUVFIREFoVFlXNGdTbTl6WlRFUgpNQThHQTFVRUNnd0lRbkp2WVdSamI
yMHhEREFLQmdOVkJBc01BMEpUVGpFak1DRUdBMVVFQXd3YVZXSjFiblIxCk1sOHpNbDh4TnpBdVluSnZ
ZMkZrWlM1amIyMHdIaGNOTWpBd01qSTNNakV6TnpVeldoY05NakV3TWpJMk1qRXoKTnpVeldqQjdNUXN
3Q1FZRFZRUUdFd0pWVXpFVE1CRUdBMVVFQ0F3S1EyRnNhV1p2Y201cFlURVJNQThHQTFVRQpCd3dJVTJ
GdUlFcHZjMlV4RVRBUEJnTlZCQW9NQ0VKeWIyRmtZMjl0TVF3d0NnWURWUVFMREFOQ1UwNHhJekFoCkJ
nTlZCQU1NR2xWaWRXNTBkVEpmTXpKZk1UY3dMbUp5YjJOaFpHVXVZMjl0TUlJQklqQU5CZ2txaGtpRzl
3MEIKQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBd2pWR09haG5Fei8zdWZnOUM5U25PVDZFZ095M2tUQ1N
jRXR1YmVXMwpHVmc1TzZONmxBaktPa2FLRjhoUytXeHBKanRQRW52TWtqNDh0Wkw1dHM4SUJrSnREbmt
2N1Njd0FHanRtVk42CmtWemRsZWY1RURFUzVwU3A4WVdOMkhNa285c0lqVmxxaVpwQTc0dlQvdzhDS2N
```

```
DaVNJdmRSeW45bUo4ejUrcjUKeEVRLzFNWWozMzZyNlE3dy9DMjhwMTVoMUR5RUo2ZTBoQ1BhRWdtYmo
3TmVzdFBRWGRpT1JzUWdOT0xySFQySgp3QjBiL0IxOWVCdUpnRXBJbzlXaTBZOVlXWjExWDhLdEszUWs
3aVE2WlltSVNJaC8vMGhyaEV6NWdVbnk5Q0R3Ck1tb1J6Wjk4K3JuUGlhR1hOT0JTdE1YbXE1VWgxdUE
vR2dkY3FMOFZCR2crTlFJREFRQUJvelV3TXpBTUJnTlYKSFJNQkFmOEVBakFBTUE0R0ExVWREd0VCL3d
RRUF3SUZvREFUQmdOVkhTVUVEREFLQmdnckJnRUZCUWNEQVRBTgpCZ2txaGtpRzl3MEJBUXNGQUFPQ0F
RRUFpVU1UT3Z2MnRTSkM3MHk4bEhaZVByaEtGdFRuR3hSNHlEUzUzQTFpCnRUN1lFNy9OaTY1ZGFiRmV
LaDZwbnhRTGM4RVFzYnFFdmNueVAxbVM4dzRYNzFrU2o0Mk5NS1QwNjNnbzhmWTMKU1EyRWh2Y0paVUx
IdEtWODFOSVZBdnBqazFVTXdGWTRuN1YrcmFWVEMyeXk1N1NFUWJRN1U0S1JScXRIbTdwRQpsUncwelh
3eHRkMG9GeTFkMEhPMjdDeDY0N3FXMDQyZ3ZSMjVrSzJuN05SQUkwdVpLVzMxRmNnell3MGJyemdiClB
oVE9QeWdGbmtIMnZFZHU1RTA2SkRuLzd6emdpNndHK09LL0dSVlZpZThTS3FrYTI4WXhCemlqWFZjVU5
GQTgKL2tnWjNDbFl5anZtSURoc1hYWnNWUTdQbWhvWURDNEV1NEl1TzU4VHBaY09ZUT09Ci0tLS0tRU5
EIENFUlRJRklDQVRFLS0tLS0=",
   "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
       }
   ]
}
```


5.6.5. Get the Registered Northbound Server

The following sample request and response uses a GET request to get the registered northbound server.

Request Headers

- Authorization = Session ID (authorization key from login API response data)

- Accept = application/json
- Content-Type = application/json

URI Structure

```
GET https://<SANnav IP>/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 body similar to the following, and a “200 OK” status in the header.


```
{
    "name": "nb_server",
    "kafkaClusterUrl": "10.24.24.25:9093",
    "schemaRegistryUrl": "http://10.24.24.25:8081",
    "caPublicCertificate":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURyakNDQXBhZ0F3SUJBZ0lKQU5oY3FZRnluOG9
wTUEwR0NTcUdTSWIzRFFFQkN3VUFNSHN4Q3pBSkJnTlYKQkFZVEFsVlRNUk13RVFZRFZRUUlEQXBEWVd
4cFptOXlibWxoTVJFd0R3WURWUVFIREFoVFlXNGdTbTl6WlRFUgpNQThHQTFVRUNnd0lRbkp2WVdSamI
yMHhEREFLQmdOVkJBc01BMEpUVGpFak1DRUdBMVVFQXd3YVZXSjFiblIxCk1sOHpNbDh4TnpBdVluSnZ
ZMkZrWlM1amIyMHdIaGNOTWpBd01qSTNNakV6TnpVeldoY05NakV3TWpJMk1qRXoKTnpVeldqQjdNUXN
3Q1FZRFZRUUdFd0pWVXpFVE1CRUdBMVVFQ0F3S1EyRnNhV1p2Y201cFlURVJNQThHQTFVRQpCd3dJVTJ
GdUlFcHZjMlV4RVRBUEJnTlZCQW9NQ0VKeWtZMjl0TVF3d0NnWURWUVFMREFOQ1UwNHhJekFoCkJnTlZ
CQU1NR2xWaWRXNTBkVEpmTXpKZk1UY3dMbUp5YjJOaFpHVXVZMjl0TUlJQklqQU5CZ2txaGtpRzl3MEI
KQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBd2pWR09haG5Fei8zdWZnOUM5U25PVDZFZ095M2tUQ1NjRXR
1YmVXMwpHVmc1TzZONmxBaktPa2FLRjhoUytXeHBKanRQRW52TWtqNDh0Wkw1dHM4SUJrSnREbmt2N1N
jd0FHanRtVk42CmtWemRsZWY1RURFUzVwU3A4WVdOMkhNa285c0lqVmxxaVpwQTc0dlQvdzhDS2NDaVN
JdmRSeW45bUo4ejUrcjUKeEVRLzFNWWozMzZyNlE3dy9DMjhwMTVoMUR5RUo2ZTBoQ1BhRWdtYmo3TmV
zdFBRWGRpT1JzUWdOT0xySFQySgp3QjBiL0IxOWVCdUpnRXBJbzlXaTBZOVlXWjExWDhLdEszUWs3aVE
2WlltSVNJaC8vMGhyaEV6NWdVbnk5Q0R3Ck1tb1J6Wjk4K3JuUGlhR1hOT0JTdE1YbXE1VWgxdUEvR2d
kY3FMOFZCR2crTlFJREFRQUJvelV3TXpBTUJnTlYKSFJNQkFmOEVBakFBTUE0R0ExVWREd0VCL3dRRUF
3SUZvREFUQmdOVkhTVUVEREFLQmdnckJnRUZCUWNEQVRBTgpCZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUF
pVU1UT3Z2MnRTSkM3MHk4bEhaZVByaEtGdFRuR3hSNHlEUzUzQTFpCnRUN1lFNy9OaTY1ZGFiRmVLaDZ
wbnhRTGM4RVFzYnFFdmNueVAxbVM4dzRYNzFrU2o0Mk5NS1QwNjNnbzhmWTMKU1EyRWh2Y0paVUxIdEt
WODFOSVZBdnBqazFVTXdGWTRuN1YrcmFWVEMyeXk1N1NFUWJRN1U0S1JScXRIbTdwRQpsUncwelh3eHR
kMG9GeTFkMEhPMjdDeDY0N3FXMDQyZ3ZSMjVrSzJuN05SQUkwdVpLVzMxRmNnell3MGJyemdiClBoVE9
QeWdGbmtIMnZFZHU1RTA2SkRuLzd6emdpNndHK09LL0dSVlZpZThTS3FrYTI4WXhCemlqWFZjVU5GQTg
KL2tnWjNDbFl5anZtSURoc1hYWnNWUTdQbWhvWURDNEV1NEl1TzU4VHBaY09ZUT09Ci0tLS0tRU5EIEN
FUlRJRklDQVRFLS0tLS0=",
    "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
        }
    ]
}
```

5.6.6. Enable or Disable a Stream

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

Request Headers

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

URI Structure

```
PUT https://<SANnav IP>/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

For enabling stream:

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

For disabling stream:

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

Response Data

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

5.6.7. Remove the Northbound Server

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

Request Headers

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

URI Structure

```
DELETE https://<SANnav IP>/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” displayed in the body header.

5.7. Python Examples

This section provides a couple of 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>
```

5.7.1. 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)
```

5.7.2. Retrieving a List of Events

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

```
import requests
import json
import time
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(logout.status_code)


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


def filterpayload(category_type, excludedEvents, includedEvents):
    payload = {}
    filter = []
    filter_json = {}
    categories = {}
    if len(category_type) > 0:
        for type in category_type:
            for k, v in type.items():
                categories[k] = v
    filter_json['categories'] = categories
    filter_json['excludedEvents'] = excludedEvents
    filter_json['includedEvents'] = includedEvents
    filter.append(filter_json)

    payload = {'filter': filter}
    return payload


def getevents(sannav_ip=None, session_id=None, header=None, startTime=None,
endTime=None, filters=None, pageSize=500,
              startIndex=0, category_type=[], excludedEvents=[],
includedEvents=[]):
    payload = eventpayload(startTime=startTime, endTime=endTime,
pageSize=pageSize, startIndex=startIndex,
                 category_type=category_type,
excludedEvents=excludedEvents,
                           includedEvents=includedEvents)
```

```
    events = requests.post(url='https://{0}/external-
api/v1/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__':
    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
    endtime = int(time.time())
    starttime = endtime - 7200

    ###Fetching the session id & header####
    session_id = login(username=username, password=password,
sannav_ip=sannav_ip)
    header = {'Authorization': session_id['sessionId'], 'Content-type':
'application/json'}

    #####Fetching the events####
    try:
        all_events = getevents(sannav_ip=sannav_ip,
session_id=session_id, header=header,
                               startTime='{0}000'.format(starttime),
endTime='{0}000'.format(endtime))
        print('All Events: {0}'.format(all_events))
    except Exception as e:
        print('Exception :{0}'.format(e))

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

Chapter 6. Revision History

6.1. SANnav-211x-REST-API-RM100; 18 December 2020

- Added Gen 7 platform support to the "Overview" and "Northbound Streaming" chapters.
- Renamed the "Examples" chapter to "Sample Requests and Responses."
- Added the "Python Examples" section to the "Sample Requests and Responses" chapter.