Cisco Security Packet Analyzer REST API Guide, 6.2(2)

Table of Contents

Cisco Security Packet Analyzer API Team <secpa-api-support@cisco.com>
Document Version 0.9-pre (2016-10-21)

Copyright © 2013-2016, Cisco Systems, Inc.

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco’s trademarks can be found at http://www.cisco.com/go/trademarks. Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1005R)

Introduction

Overview

The Cisco Security Packet Analyzer (SECPA) REST API Guide describes the programming interfaces available for configuring and retrieving data from the SECPA. The SECPA API follows the commonly-used Representational State Transfer (REST) style of providing services over HTTP or HTTPS. The eXtensible Markup Language (XML) is used for both requests sent to and responses returned by the SECPA.

The REST API can be used to configure a large subset of the SECPA's software features, including sites, data sources, applications, application groups, thresholds, alarm actions, packet capture, and system information.

This release of the REST API Guide covers up to version 6.2(2) of the SECPA software.

Note that some parts of this document refer to the term "NBI" (Northbound Interface). While this term is not unusual in the networking field, it is not generally familiar to software developers at large. Therefore, while the "NBI" term will be mostly avoided, it should be understood that in the context of this document, "NBI" is synonymous with "REST API".

Audience

This guide is intended to be a technical resource for application developers who want to use the REST API to configure the SECPA and leverage its data programatically. Developers should be familiar with a high-level programming language such as Java, C#, Python, or similar. Additionally, some knowledge of the following is recommended:

- Hypertext Transfer Protocol (HTTP/HTTPS)
- XML and XML Schema
- PHP: Hypertext Preprocessor
- Structured Query Language (SQL)

Developers should also be familiar with the SECPA GUI and have a basic understanding of SECPA functionality, as in most cases, API operations correspond closely to GUI operations.

Document Formats

This guide is available in HTML and PDF formats:

- HTML — https://cisco-secpa.github.io/rest-api-guide/
- PDF — https://cisco-secpa.github.io/rest-api-guide/secpa-rest-api-guide.pdf

Related Documentation

Cisco Security Packet Analyzer User Guide

http://www.cisco.com/c/en/us/support/security/security-packet-analyzer/products-user-guide-list.html

Cisco Security Packet Analyzer Install and Upgrade Guides

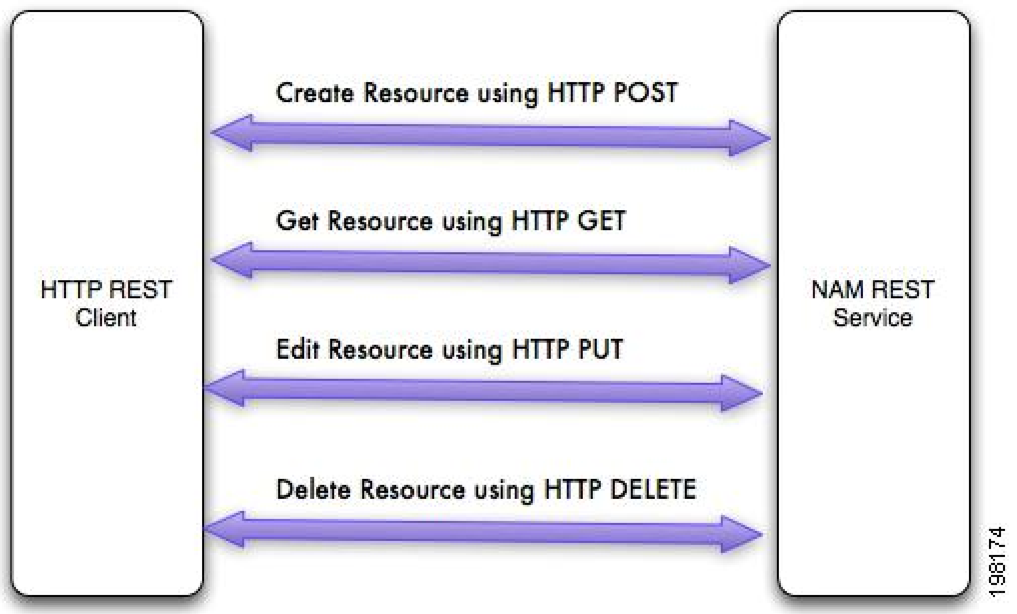
http://www.cisco.com/c/en/us/support/security/security-packet-analyzer/products-installation-guides-list.html

REST API

Overview

The REST architectural style relies on standard HTTP methods (such as GET, POST, PUT, and DELETE) to execute web service requests. In the SECPA API, operations can be broadly divided into four types, with each type generally mapping to one of the HTTP methods. Each call for an API operation involves a client request (which consists of a request URI and possibly a request body) and a server response (which consists of an HTTP response code and a response body). Request and response bodies are in XML format.

The figure below illustrates the interactions between a REST client and the SECPA REST API.


HTTP REST Client
Create Resource using HTTP POST
Get Resource using HTTP GET
Edit Resource using HTTP PUT
Delete Resource using HTTP DELETE
NAM REST Service
198174

NAM REST Service interaction diagram

The list below summarizes the elements of client requests and server responses.

- Request URI — This specifies the object to be acted upon (except when a new object is to be created).
- Request body — This specifies additional parameters that may be needed to process the request.
- Response code — This is the standard HTTP response code; it indicates whether a request was successful overall.
- Response body — This includes a SECPA-specific status code and description, and may include additional information.

The table below shows the mapping between operation types and HTTP methods, and describes the general form of request and response bodies. Refer to REST API Reference for details on each specific API operation, including the format of the request URI, request body, and response body.

Operation Type	HTTP Method	Description
Create	POST	Creates an object. A request body specifying the object’s parameters is required. The response body includes the new object’s ID.
Read	GET or POST	Reads out a list of objects, or details of a specific object. Generally a GET is used (so no request body is required), unless the read parameters cannot be easily encoded in the URI, in which case a POST is used, with the parameters sent in the request body. The response body includes the list or details requested.
Update	PUT	Updates an object. A request body describing the object’s parameters is required. The response body includes only the minimum required.
Delete	DELETE	Deletes an object. No request body is required. The response body includes only the minimum required.

The following is an example of a response body:

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
    <uri>/nbi/nbi-site/id/9</uri>
  </operation-result>
</nam-response>
```

where:

- **status** indicates whether the operation is successful (and if not, provides a more specific reason than the HTTP response code).
- **description** is a human-readable description of the status code.
- **uri** specifies a new object’s identifier, which may be used to refer to the object in future API calls.

XML Schema

XML Schema is a standard defined by the World Wide Web Consortium (W3C). Further details about the standard can be found at http://www.w3.org/XML/Schema.

The SECPA software uses an XML schema to validate that an XML request received from the client is syntactically well-formed. The validation process checks that XML elements in the request are recognized, and that various constraints relevant to the request type are satisfied. However, note that the validation cannot be relied upon to detect all input errors, as some types of constraints are difficult, or even impossible, to express in the XML Schema language. Nevertheless, developers may still find it useful to refer to the XML schema file as an additional resource to help properly format XML requests to SECPA. The schema file is included with the SECPA image; it can be downloaded at http://`secpa-host`/admin/nbischema.php, where `secpa-host` denotes the hostname or IP address of the SECPA.

Note that in XML Schema, the order of child elements under an `<xs:sequence>` element is significant. Therefore, verify that the XML request elements are in the correct order if the API returns an error message like

```
Error 1871: Element 'foo': This element is not expected. Expected is one of ( bar, baz
). on line 6
```

Development Tools

When developing with the SECPA REST API (or in general, any REST API), you will likely find that a good REST client is an indispensable tool for testing and experimentation. There are many REST client options available, and you are free to use any client you prefer. However, if you have no particular preferences, then we recommend the following:

- Google Chrome: Postman (http://www.getpostman.com/)
- Mozilla Firefox: RESTClient (http://restclient.net/)

Authentication

To use the APIs, the client program must obtain a session ID from the SECPA. A successful login request returns a session ID that remains valid until either a logout request is received, or 30 minutes of inactivity has elapsed (the inactivity interval is not currently user-configurable). A session ID can be used to make as many API requests as desired, and is tied to the IP address that originated the login (to prevent session hijacking). An API request without a valid session ID is rejected.

A Python-based reference implementation of the authentication steps below can be found at:

> https://github.com/cisco-secpa/rest-api-auth/

The following steps outline the procedure to log in to the SECPA server:

1. Perform an HTTP GET from

   > http://`secpa-host`/auth/login.php?api=true

   where `secpa-host` is the hostname or IP address of the SECPA server.

   This returns the following key/value pairs:

   - `domain` — a string representing the base URL of the SECPA web server.
   - `nonce` — the hex representation of a random number, used for security purposes.

  - `pkey` — the hex representation of the SECPA server's public key.
  - `sessid` — the hex representation of the PHP session ID to be included with future API requests.

2. Encode the password using the appropriate method below.

  - If the SECPA is configured to authenticate local users only, then `nonce` will be non-null. In this case, generate an MD5 hash of the password as follows:

    ```
    encoded_pw = MD5(domain + nonce + username + password_hash)
    ```

    where:

    - + denotes string concatenation
    - `password_hash` = **SHA-1**(`salt`, `username`, `password`)
    - `salt` = "04581273"

    Note that `nonce` and `password_hash` are both ASCII strings of hex digits.

  - If the SECPA is configured to use TACACS+ authentication, then `nonce` will be null. In this case, encode the password using the following algorithm (based on the Diffie-Hellman key exchange):

    ```
    privKey = random()
    pubKey = g^privKey mod m
    shared = MD5(pkey^privKey mod m)
    encoded_pw = one_pass(password, shared)
    ```

    where:

    - `g` = `0x527d44089958ca1e` (generator)
    - `m` = `0x5c13ada6c91d2ba3` (modulus)
    - `pkey` is the server's public key (returned during the initial authentication request)
    - **random** is a random number generation function
    - **one_pass** is a string XOR function

3. Send the encoded password to the SECPA at the following URL:

    ```
    http://secpa-host/auth/authenticate.php?sessid=sessid&username=user&pwdigest=
    pw&pkey=pk
    ```

    where

  - `secpa` is the hostname or IP address of the SECPA server.

- `sessid` is the session ID (returned during the initial authentication request).
- `user` is the username to log in with.
- `pw` is the encoded password (encoded using the appropriate method, as described in the previous step)
- `pk` is the pubKey value calculated in the TACACS+ case (omit or set to 0 for local authentication)

If login is successful, the SECPA returns an HTTP response with the string “success”; otherwise, the SECPA returns “fail”.

Subsequent requests to the SECPA should include the HTTP cookie `PHPSESSID`. This can be accomplished by including with the request an HTTP header of the form:

```
Cookie: PHPSESSID=sessid
```

A REST API session can be explicitly terminated by visiting the following URL:

```
http://secpa-host/auth/logout.php
```

REST API Reference

Sites

A site is a grouping of hosts (network endpoints) that should be treated as a single unit in order to simplify traffic monitoring and troubleshooting.

Each site is defined by one or more rules. Each rule specifies a subnet and, optionally, a data source. A site definition might specify multiple rules in order to create an aggregated view of all network traffic across multiple data sources. When defining a site using multiple data sources, be careful to ensure that those data sources do not have duplicated traffic, or else double-counting may occur in site traffic statistics.

The Data Sources API can be used to look up any data source IDs needed to define a site.

For more information on sites, see the "Configuring Sites" section of SECPA User Guide.

API Overview

URL	Method	Description
/nbi/nbi-site	POST	Create a site.
/nbi/nbi-site	GET	List all sites.
/nbi/nbi-site/id/`id`	GET	List configuration details of a site.
/nbi/nbi-site/id/`id`	PUT	Update a site.
/nbi/nbi-site/id/`id`	DELETE	Delete a site.

Create Site

Method	POST
URI	/nbi/nbi-site
Description	Create a site.
HTTP Normal Response Code(s)	created(201)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-1), maxSites(-2), maxRules(-3), maxPrefixes(-4), maxName(-5), alreadyExists(-6), invalidID(-7), badParameters(-8), internalError(-9), noMoreData(-10), duplicateName(-11)

Only one site can be created at a time. A maximum of 1024 sites, 16 datasources, 16 prefixes, and 16 rules can be created.

Sample Request

Refer to Site Elements for additional details on specific elements below.

```
<sites>
  <site>
    <name>lab</name>
    <description>Lab</description>
    <status>0</status>
    <siteRules>
      <rule>
        <prefix>
          <ip>172.20.0.0</ip>
          <netmask>16</netmask>
        </prefix>
        <dataSource>
          <id>1</id>
        </dataSource>
      </rule>
      <rule>
        <prefix>
          <ip>172.21.0.0</ip>
          <netmask>16</netmask>
        </prefix>
        <dataSource>
          <id>2</id>
        </dataSource>
      </rule>
    </siteRules>
  </site>
</sites>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
    <uri>/nbi/nbi-site/id/2</uri>
  </operation-result>
</nam-response>
```

Site Elements

Element	Description
id	Uniquely identifies the resource. This value should be saved for future references to the specific site. For example, in the XML response above, the unique site ID is 2.
name	The name of the resource.
description	A textual description of the resource.
prefix	The subnet address prefix and subnet mask.
dataSource	The (optional) data source referenced by this rule. Data sources can be retrieved using the Data Sources API.

List Sites

Method	GET
URI	/nbi/nbi-site
Description	List all sites.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), internalError(500)
SECPA Status Code(s)	successful(0), resourceError(-1), internalError(-9)

Sample Request

This operation does not require a request body.

Sample Response

Refer to Site Elements for additional details on specific elements below.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <sites>
    <site id="1">
      <name>Unassigned</name>
      <description>Unassigned hosts</description>
      <status>enable</status>
    </site>
    <site id="2">
      <name>lab</name>
      <description>Lab</description>
      <status>enable</status>
    </site>
  </sites>
</nam-response>
```

List Site Details

Method	GET
URI	/nbi/nbi-site/id/`id`
Description	List configuration details of a site.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), resourceError(-1), invalid(-7), badParameter(-8), internalError(-9)

Sample Request

This operation does not require a request body.

Sample Response

Refer to Site Elements for additional details on specific elements below.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <site id="2">
    <name>lab</name>
    <description>Lab</description>
    <status>enable</status>
    <siteRules>
      <rule>
        <prefix>
          <ip>172.20.0.0</ip>
          <netmask>16</netmask>
        </prefix>
        <dataSource>
          <id>1</id>
          <name>DATA PORT 1</name>
        </dataSource>
      </rule>
      <rule>
        <prefix>
          <ip>172.21.0.0</ip>
          <netmask>16</netmask>
        </prefix>
        <dataSource>
          <id>2</id>
          <name>DATA PORT 2</name>
        </dataSource>
      </rule>
    </siteRules>
  </site>
</nam-response>
```

Update Site

Method	PUT
URI	/nbi/nbi-site/id/`id`
Description	Update a site.
HTTP Normal Response Code(s)	created(201)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-1), maxRules(-3), maxPrefixes(-4), maxName(-5), alreadyExists(-6), invalidID(-7), badParameters(-8), internalError(-9), noMoreData(-10), duplicateName(-11)

Sample Request

Refer to Site Elements for additional details on specific elements below.

```
<sites>
  <site>
    <name>lab</name>
    <description>Lab</description>
    <status>0</status>
    <siteRules>
      <rule>
        <prefix>
          <ip>172.20.0.0</ip>
          <netmask>16</netmask>
        </prefix>
      </rule>
      <rule>
        <prefix>
          <ip>172.21.0.0</ip>
          <netmask>16</netmask>
        </prefix>
      </rule>
    </siteRules>
  </site>
</sites>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Delete Site

Method	DELETE
URI	/nbi/nbi-site/id/`id`
Description	Delete a site.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)

SECPA Status Code(s)	successful(0), resourceError(-1), invalidID(-7), badParams(-8), internalError(-9)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Data Sources

Data sources provide traffic to the SECPA software. They are primarily used to define sites. Some examples of data sources are:

- A physical data port on the SECPA that receives SPAN data.
- A router or switch that sends NetFlow data to the SECPA.
- An ERSPAN session with a specific ERSPAN ID.

For more information on data sources, see the "Data Source Overview" section of SECPA User Guide.

API Overview

URL	Method	Description
/nbi/nbi-datasource	GET	List all data sources.
/nbi/nbi-datasource/id/id	GET	List configuration details of a data source.

List Data Sources

Method	GET
URI	/nbi/nbi-datasource
Description	List all data sources.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), internalError(500), notImplemented(501)

SECPA Status Code(s)	successful(0), resourceError(-1)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <dataSources>
    <dataSource>
      <id>1</id>
      <description>DATA PORT 1</description>
      <dsrcType>1</dsrcType>
    </dataSource>
    <dataSource>
      <id>2</id>
      <description>DATA PORT 2</description>
      <dsrcType>1</dsrcType>
    </dataSource>
  </dataSources>
</nam-response>
```

List Data Source Details

Method	GET
URI	/nbi/nbi-datasource/id/`id`
Description	List configuration details of a data source.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), resourceError(-1)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <dataSources>
    <dataSource>
      <id>1</id>
      <description>DATA PORT 1</description>
      <dsrcType>1</dsrcType>
    </dataSource>
  </dataSources>
</nam-response>
```

Applications

The SECPA implements an application classification system that associates traffic with a particular application by matching it against a database of rules. These rules can be categorized into three types:

- Protocol, which matches TCP or UDP traffic on a given port or port range.
- HTTP URL, which matches HTTP traffic associated with a particular domain and/or path.
- Server IP Address, which matches traffic associated with a particular IP address, IP range, or subnet (and optionally, port or port range).

Each application is uniquely identified by an **application tag**, or **apptag** for short. (In some contexts, the apptag may also be referred to as an **application ID**.) The apptag itself is derived from an **engine ID** and a **selector**. The engine ID denotes a repository or namespace of application definitions (e.g., IANA layer 3 protocols, or IANA layer 4 port numbers), while the selector uniquely identifies an application within an engine ID (e.g., port 80 is mapped to selector 80 in IANA-l4). Note, however, that the REST API uses only the apptag to reference a particular application; the engine ID and selector are not used, and are mentioned here only for completeness.

The SECPA includes a set of pre-defined applications, which cannot be modified or deleted. The SECPA also allows custom applications to be defined. Each custom application is defined by a set of rules (of the three types described previously).

For more information on applications, see the "Configuring Application Classification" section of SECPA User Guide.

API Overview

URL	Method	Description
/nbi/nbi-apps	POST	Create a custom application.
/nbi/nbi-apps	GET	List all applications (pre-defined and custom).

URL	Method	Description
/nbi/nbi-apps/apptag/`tag`	GET	List configuration details of an application.
/nbi/nbi-apps/apptag/`tag`	PUT	Update a custom application. (Pre-defined applications cannot be updated.)
/nbi/nbi-apps/apptag/`tag`	DELETE	Delete a custom application. (Pre-defined applications cannot be deleted.)

Create Custom Application

Method	POST
URI	/nbi/nbi-apps
Description	Create a custom application. Valid rule elements are: `match`, for rules based on protocol (TCP/UDP) and port number only (no IP address). `ipmatch`, for rules based on some combination of IP address, protocol, and port number. `urlMatch`, for rules based on HTTP URL (host and path). If an `ipmatch` rule is used, then a `type` element is used to specify the type of IP match to be performed. Valid `type` elements are: `serverip`, for matching against a single IP address (e.g., `10.0.0.1`). `serverrange`, for matching against an IP address range (e.g., `10.0.0.1-10.0.0.15`). `subnet`, for matching against an IP subnet (e.g., `10.0.0.0/24`).
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	Success (0), No Such Application (-1), Resource Error (-2), RPC Connection Error (-3), Bad Arguments (-4), Access Error (-5), Application Exists (-6), Maxium Matches Reached (-7), Invalid Engine Id (-8), Invalid Selector (-9), Invalid Parent (-10), Invalid Range (-11), Invalid Cookie (-12), Match Exists (-13), Server Overlap (-15), Port Overlap (-16), Invalid Protocol (-17), Invalid Server (-18), Invalid Port (-19), Invalid Regex (-20)

Sample Request

Protocol:

```xml
<applicationId>
  <name>Protocol-UDP</name>
  <description>Protocol-based</description>
  <matches>
    <!-- More than one <match> element can be specified here, if desired. -->
    <match>
      <layer>UDP</layer>
      <ports>1000-1099</ports>
    </match>
  </matches>
</applicationId>
```

HTTP URL (e.g., http://host.domain.com/intro?id=123):

```xml
<applicationId>
  <name>HTTP-URL</name>
  <description>HTTP URL-based</description>
  <matches>
    <urlMatch>
      <!--
          Note that each match element below is treated as a regular expression,
          so metacharacters must be escaped accordingly (e.g., '?' becomes '\?').
      -->
      <host>host\.domain\.com</host>
      <path>/intro\?id=123</path>
    </urlMatch>
  </matches>
</applicationId>
```

Server IP Address:

```
<applicationId>
   <name>IP-ADDRESS</name>
   <description>Several IP address rules</description>
   <matches>
     <!--
         Examples of the different types of IP address matching.
         Any combination of these types of IP address matches can
         be used in a single application definition.
     -->
     <ipMatch>
       <type>serverip</type>
       <address>10.0.0.1</address>
       <protocol>TCP</protocol>
       <ports>80</ports>
     </ipMatch>
     <ipMatch>
       <type>serverrange</type>
       <address>10.0.1.1-10.0.1.15</address>
       <protocol>TCP</protocol>
       <ports>8000-8100</ports>
     </ipMatch>
     <ipMatch>
       <type>subnet</type>
       <address>10.0.2.0/24</address>
       <protocol>TCP</protocol>
     </ipMatch>
   </matches>
</applicationId>
```

Multiple Rules:

```
<applicationId>
  <name>MultipleRule</name>
  <description>Example of multiple rule types in a single definition</description>
  <matches>
    <match>
      <layer>UDP</layer>
      <ports>2000-2099</ports>
    </match>
    <ipMatch>
      <type>serverip</type>
      <address>10.10.10.10</address>
      <protocol>TCP</protocol>
      <ports>567</ports>
    </ipMatch>
    <urlMatch>
      <host>host\.domain\.com</host>
      <path>/intro</path>
    </urlMatch>
  </matches>
</applicationId>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
    <uri>/nbi/nbi-apps/apptag/268435458</uri>
  </operation-result>
</nam-response>
```

List Applications

Method	GET
URI	/nbi/nbi-apps
Description	List all applications (pre-defined and custom).
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), noSuchApp(-1), resourceError(-2), internalError(-3), badParameter(-4), invalidEngineId(-8)

Sample Request

This operation does not require a request body.

Sample Response

```
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <applicationId>
    <name>icmp</name>
    <appTag>16777217</appTag>
    <engineId>iana-l3</engineId>
    <selector>1</selector>
    <matches>
      <match>
        <layer> IP</layer>
        <ports>1</ports>
      </match>
    </matches>
  </applicationId>
</nam-response>
```

List Application Details

Method	GET
URI	/nbi/nbi-apps/apptag/`tag`
Description	List configuration details of an application.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), noSuchApp(-1), resourceError(-2), internalError(-3), badParameter(-4), invalidEngineId(-8)

Sample Request

This operation does not require a request body.

Sample Response

```
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <applicationId>
    <name>icmp</name>
    <appTag>16777217</appTag>
    <engineId>iana-l3</engineId>
    <selector>1</selector>
    <matches>
      <match>
        <layer>IP</layer>
        <ports>1</ports>
      </match>
    </matches>
  </applicationId>
</nam-response>
```

Update Custom Application

Method	PUT
URI	/nbi/nbi-apps/apptag/`tag`
Description	Update a custom application. (Pre-defined applications cannot be updated). Refer to Create Custom Application for details on formatting the request body.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	Success (0), No Such Application (-1), Resource Error (-2), RPC Connection Error (-3), Bad Arguments (-4), Access Error (-5), Application Exists (-6), Maxium Matches Reached (-7), Invalid Engine Id (-8), Invalid Selector (-9), Invalid Parent (-10), Invalid Range (-11), Invalid Cookie (-12), Match Exists (-13), Server Overlap (-15), Port Overlap (-16), Invalid Protocol (-17), Invalid Server (-18), Invalid Port (-19), Invalid Regex (-20)

Sample Request

```
<applicationId>
  <name>NDE-TCP-9999</name>
  <matches>
    <match>
      <layer>TCP</layer>
      <ports>9999</ports>
    </match>
  </matches>
</applicationId>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
    <uri>/nbi/nbi-apps/apptag/268435458</uri>
  </operation-result>
</nam-response>
```

Delete Custom Application

Method	DELETE
URI	/nbi/nbi-apps/apptag/`tag`
Description	Delete a custom application. (Pre-defined applications cannot be deleted.)
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), noSuchApp(-1), resourceError(-2), internalError(-3), badParameter(-4), invalidEngineId(-8)

Sample Request

This operation does not require a request body.

Sample Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Application Setup

SECPA supports two engines for processing packet flows and classifying them as specific applications:

- The "classic" engine, which classifies flows based on IP protocol and port number.
- The NBAR2 engine, which performs Deep Packet Inspection (DPI) for more accurate classification. The DPI functionality relies on a **protocol pack**, which encodes the rules that determine how packet flows are classified. Each SECPA software image includes a built-in, **default** protocol pack. From time to time, an updated protocol pack should be uploaded to the SECPA in order to reflect changes to existing protocols and the addition of new protocols. In case of problems with an update, the default protocol pack always remains available for use.

The Application Setup API is used to manage protocol packs, as well as to control whether DPI (i.e., the NBAR2 engine) is enabled. If DPI is not enabled, then the "classic" engine is used. While the NBAR2 engine must perform more detailed packet analysis, this generally does not have a significant impact on monitored traffic throughput. Thus, DPI is enabled by default, but users who find that they are better served by the default engine may choose to disable DPI functionality.

API Overview

URL	Method	Description
/nbi/nbi-appsetup/protocolpack	POST	Upload a new protocol pack and set it as the current protocol pack.
/nbi/nbi-appsetup/protocolpack	GET	List details of the current and default protocol packs.
/nbi/nbi-appsetup/protocolpack	DELETE	Delete the current protocol pack and revert to the default protocol pack.
/nbi/nbi-appsetup/dpisetup	GET	Get the Deep Packet Inspection configuration.
/nbi/nbi-appsetup/dpisetup	POST	Set the Deep Packet Inspection configuration.

Upload Protocol Pack

Method	POST
URI	/nbi/nbi-appsetup/protocolpack

Description	Upload a new protocol pack and set it as the current protocol pack.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0),failed(1),not supported(2),invalid-url(3)

Sample Request

Refer to Protocol Pack Request Elements for additional details on specific elements below.

```
<applicationSetup>
  <protocolPack>
    <url>http://hostname/filepath/xyz.pack</url>
    <username>username</username>  <!-- optional -->
    <password>password</password>  <!-- required whenever a username is provided -->
  </protocolPack>
</applicationSetup>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Success</description>
  </operation-result>
</nam-response>
```

Protocol Pack Request Elements

Element	Description
url	The URL of the protocol pack file (supported schemes: ftp, http, https, scp, sftp).
username	The username for accessing the file (optional).
password	The password for accessing the file (required whenever a username is provided).

List Protocol Pack Details

Method	GET
URI	/nbi/nbi-appsetup/protocolpack

Description	List details of the current and default protocol packs (version, engine version, description, etc.).
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), internalError(500)
SECPA Status Code(s)	successful(0), resourceError(-1), internalError(-9)

Sample Request

This operation does not require a request body.

Sample Response

Refer to Protocol Pack Response Elements for additional details on specific elements below.

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <id>0</id>
  <protocolPack>Current</protocolPack>
  <description>Advanced Protocol Pack</description>
  <version>9.01</version>
  <engVersion>18</engVersion>
  <id>1</id>
  <protocolPack>Default</protocolPack>
  <description>Advanced Protocol Pack</description>
  <version>7.1</version>
  <engVersion>18</engVersion>
</nam-response>
```

Protocol Pack Response Elements

Element	Description
id	Uniquely identifies the protocol pack.
protocolPack	The protocol pack label ("Current" or "Default").
description	A textual description of the protocol pack.
version	The protocol pack version.
engVersion	The NBAR2 engine version.

Delete Protocol Pack

Method	DELETE

URI	/nbi/nbi-appsetup/protocolpack
Description	Delete the current protocol pack and revert to the default protocol pack.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0),failed(1)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Restored Default Protocol Pack</description>
  </operation-result>
</nam-response>
```

Get Deep Packet Inspection Configuration

Method	GET
URI	/nbi/nbi-appsetup/dpisetup
Description	Get the Deep Packet Inspection configuration. Currently, this includes only the enabled/disabled state.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), internalError(500)
SECPA Status Code(s)	successful(0), resourceError(-1), internalError(-9)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <applicationSetup>
    <dpi>enable</dpi>
  </applicationSetup>
</nam-response>
```

Set Deep Packet Inspection Configuration

Method	POST
URI	/nbi/nbi-appsetup/dpisetup
Description	Set the Deep Packet Inspection configuration. Currently, this includes only the enabled/disabled state.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0),failed(1),not supported(2),invalid-url(3),Xml-Body-Empty(4)

Sample Request

```
<applicationSetup>
  <dpi>disabled</dpi>
</applicationSetup>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Success</description>
  </operation-result>
</nam-response>
```

Application Groups

An **application group** is a set of application protocols that can be monitored as a unit. For example, all protocols related to file transfers could be added to a `File-Transfers` group for more convenient monitoring of that type of traffic.

For more information on application groups, see the "Configuring Application Groups" section of SECPA User Guide.

API Overview

URL	Method	Description
/nbi/nbi-appgrp	POST	Create an application group.
/nbi/nbi-appgrp	GET	List all application groups.
/nbi/nbi-appgrp/id/`id`	GET	List configuration details of an application group.
/nbi/nbi-appgrp/id/`id`	PUT	Update an application group.
/nbi/nbi-appgrp/id/`id`	DELETE	Delete an application group.

Create Application Group

Method	POST
URI	/nbi/nbi-appgrp
Description	Create an application group.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), noSuchResource(-1), outOfResources(-2), internalError(-3), badParameter(-4), accessError(-5), alreadyExists(-6), failedError(-7)

Sample Request

```xml
<appGroups>
  <appGroup>
    <name>Test-Group</name>
    <apps>
      <app>
        <name>ldap</name>
        <tag>50332037</tag>
      </app>
      <app>
        <name>ldaps</name>
        <tag>50332284</tag>
      </app>
      <app>
        <name>msft-gc</name>
        <tag>50334916</tag>
      </app>
      <app>
        <name>msft-gc-ssl</name>
        <tag>50334917</tag>
      </app>
    </apps>
  </appGroup>
</appGroups>
```

Sample Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
    <uri>/nbi/nbi-appgrp/id/41</uri>
  </operation-result>
</nam-response>
```

List Application Groups

Method	GET
URI	/nbi/nbi-appgrp
Description	List all application groups.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), noSuchResource(-1), outOfResources(-2), internalError(-3), accessError(-5), failedError(-7)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <appGroups>
    <appGroup>
      <id>1</id>
      <name>Authentication</name>
    </appGroup>
    <appGroup>
      <id>2</id>
      <name>Backup</name>
    </appGroup>
    <appGroup>
      <id>3</id>
      <name>Call-Management</name>
    </appGroup>
  </appGroups>
</nam-response>
```

List Application Group Details

Method	GET
URI	/nbi/nbi-appgrp/id/`id`
Description	List configuration details of an application group.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), noSuchResource(-1), outOfResources(-2), internalError(-3), badParameter(-4), accessError(-5), alreadyExists(-6), failedError(-7)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <appGroups>
    <appGroup>
      <id>41</id>
      <name>Test-Group</name>
      <apps>
        <app>
          <name>ldap</name>
          <tag>50332037</tag>
        </app>
        <app>
          <name>ldaps</name>
          <tag>50332284</tag>
        </app>
        <app>
          <name>msft-gc</name>
          <tag>50334916</tag>
        </app>
        <app>
          <name>msft-gc-ssl</name>
          <tag>50334917</tag>
        </app>
      </apps>
    </appGroup>
  </appGroups>
</nam-response>
```

Update Application Group

Method	PUT
URI	/nbi/nbi-appgrp/id/id
Description	Update an application group. The application group name cannot be updated.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), noSuchResource(-1), outOfResources(-2), internalError(-3), badParameter(-4), accessError-(5), alreadyExists(-6), failedError(-7)

Sample Request

```
<appGroups>
  <appGroup>
    <name>Test-Group</name> <!-- Must be the same name as before. -->
    <apps>
      <app>
        <name>ldap</name>
        <tag>50332037</tag>
      </app>
      <app>
        <name>ldaps</name>
        <tag>50332284</tag>
      </app>
    </apps>
  </appGroup>
</appGroups>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Delete Application Group

Method	DELETE
URI	/nbi/nbi-appgrp/id/`id`
Description	Delete an application group.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), noSuchResource(-1), outOfResources(-2), internalError(-3), badParameter(-4), accessError(-5), failedError(-7)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Alarm Actions

Alarm actions (or **actions** for short) are executed in response to the activation of a **threshold**. Alarm actions are used only in connection with such thresholds.

An alarm action can include any or all of the following steps:

- Send an email alert.
- Send an SNMP trap.
- Start, stop, or stop/save a packet capture session.
- Log to a remote syslog daemon.

For more information on alarm actions and thresholds, see the "Setting Up Alarms and Alarm Thresholds" section of SECPA User Guide, as well as the Thresholds API.

API Overview

URL	Method	Description
/nbi/nbi-action	POST	Create an alarm action.
/nbi/nbi-action	GET	List all alarm actions.
/nbi/nbi-action/id/`id`	GET	List configuration details of an alarm action.
/nbi/nbi-action/id/`id`	PUT	Update an alarm action.
/nbi/nbi-action/id/`id`	DELETE	Delete an alarm action.

Create Alarm Action

Method	POST
URI	/nbi/nbi-action
Description	Create an alarm action.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)

SECPA Status Code(s)	successful(0), internalError(-1), badParameter(-3), maxName(-4), NoSuchResource(-5), invalidID(-14), emailError(-15), trapCommunityError(-16), triggerCaptureError(-17), syslogErr(-18)

Sample Request

```
<alarmAction>
  <name>All actions Test</name>
  <email>enabled</email>
  <trap>
    <community>public</community>
  </trap>
  <triggerCapture>
    <session>4</session>
    <action>start</action>
  </triggerCapture>
  <syslog>enabled</syslog>
</alarmAction>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
    <uri>/nbi/nbi-action/id/5</uri>
  </operation-result>
</nam-response>
```

List Alarm Actions

Method	GET
URI	/nbi/nbi-action
Description	List all alarm actions.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), internalError(-1), badParameter(-3), NoSuchResource(-5), invalidID(-14)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <alarmAction>
    <id>6</id>
    <name>All actions</name>
    <email>enabled</email>
    <trap>
      <community>public</community>
    </trap>
    <triggerCapture>
      <session>4</session>
      <action>start</action>
    </triggerCapture>
    <syslog>enabled</syslog>
  </alarmAction>
  <alarmAction>
    <id>2</id>
    <name>email and trap actions</name>
    <email>enabled</email>
    <trap>
      <community>public</community>
    </trap>
  </alarmAction>
</nam-response>
```

List Alarm Action Details

Method	GET
URI	/nbi/nbi-action/id/`id`
Description	List configuration details of an alarm action.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), internalError(-1), badParameter(-3), NoSuchResource(-5), invalidID(-14)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <alarmAction>
    <id>3</id>
    <name>All Actions Enabled</name>
    <email>enabled</email>
    <trap>
      <community>public</community>
    </trap>
    <triggerCapture>
      <session>3</session>
      <action>start</action>
    </triggerCapture>
    <syslog>enabled</syslog>
  </alarmAction>
</nam-response>
```

Update Alarm Action

Method	PUT
URI	/nbi/nbi-action/id/`id`
Description	Update an alarm action.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), internalError(-1), badParameter(-3), maxName(-4), NoSuchResource(-5), invalidID(-14), emailError(-15), trapCommunityError(-16), triggerCaptureError(-17), syslogErr(-18)

Sample Request

```
<alarmAction>
  <name>All actions Test</name>
  <email>enabled</email>
  <triggerCapture>
    <session>4</session>
    <action>stop</action>
  </triggerCapture>
  <syslog>enabled</syslog>
</alarmAction>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Delete Alarm Action

Method	DELETE
URI	/nbi/nbi-action/id/`id`
Description	Delete an alarm action.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), internalError(-1), badParameter(-3), NoSuchResource(-5), invalidID(-14)

Sample Request

This operation does not require a request body.

Sample Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Thresholds

A **threshold** can be used to instruct the SECPA to autonomously execute specified **alarm actions** (start a packet capture, send an email alert, etc.) when a monitored condition is detected. Since the definition of a threshold references the alarm actions to be executed, those actions must be defined before the threshold itself is defined. Actions can be defined via the GUI or the Alarm Actions API.

Each threshold has a **type** associated with a particular monitoring engine. The following threshold types are supported:

- Host
- Conversation
- Application
- Response Time (formerly known as IAP, or Intelligent Application Performance)
- DSCP (DiffServ)
- RTP Streams
- Voice Signaling
- NDE Interface

Each type of threshold supports a specific set of properties. To determine the exact XML format of a threshold definition, it is often easiest to first use the GUI to create a threshold with the desired properties, and then use a REST client to list all defined thresholds.

All threshold types share the common concept of **rising** and **falling** actions. Rising actions are triggered only on a rising edge, and falling actions only on a falling edge. More specifically, during each monitoring interval, the SECPA compares the current value of the threshold's metric to its value during the previous interval, as follows:

- If the previous value was less than the rising threshold value, and the current value is now greater than the rising threshold value, then the rising threshold has been crossed (on a rising edge), so the rising action is triggered.
- If the previous value was greater than the falling threshold value, and the current value is now less than the falling threshold value, then the falling threshold has been crossed (on a falling edge), so the falling action is triggered.

A threshold may define more than one metric, in which case the comparisons are performed for each metric. Each comparison that is satisfied will trigger the corresponding action (rising or falling, as appropriate).

Note that for each metric, only one action (rising or falling) can be triggered during each monitoring interval; it is not possible for both the rising and falling actions to be triggered. Furthermore, if the previous and current values for a metric are the same, then no threshold could be crossed, so neither rising nor falling action would be taken.

For more information on thresholds and alarm actions, see the "Setting Up Alarms and Alarm Thresholds" section of SECPA User Guide, as well as the Alarm Actions API.

API Overview

URL	Method	Description
/nbi/nbi-threshold/type/`type`	POST	Create a threshold.
/nbi/nbi-threshold	GET	List all thresholds.
/nbi/nbi-threshold/id/`id`	GET	List configuration details of a threshold.
/nbi/nbi-threshold/type/`type`	PUT	Update a threshold.
/nbi/nbi-threshold/id/`id`	DELETE	Delete a threshold.

Create Threshold

Method	POST
URI	/nbi/nbi-threshold/type/`type`, where the supported threshold types are `host`, `convs`, `app`, `iap`, `dscp`, `rtp`, `voice`, and `nde`.
Description	Create a threshold.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), internalError(-1), badParameter(-3), maxName(-6), siteIDError(-8), appIDError(-9), addressError(-10), metricError(-11), codecError(-12), severityIDError(-13), actionIDError(-14), dscpError(-19), invalidID(-20), typeError(-21), ifIndexError(-22), clntSrvrError(-23)

Sample Request

```
<iapThreshold>
  <name>Response Time threshold</name>
  <severity>High</severity>
  <risingAction>1</risingAction>
  <fallingAction>1</fallingAction>
  <application>any</application>
  <server>
    <site>Any</site>
    <host>Any</host>
  </server>
  <iapMetrics>
    <risingAverageResponseTime>300</risingAverageResponseTime>
    <fallingAverageResponseTime>250</fallingAverageResponseTime>
    <risingMaxNetworkTime>100</risingMaxNetworkTime>
    <fallingMaxNetworkTime>80</fallingMaxNetworkTime>
  </iapMetrics>
</iapThreshold>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
    <uri>/nbi/nbi-threshold/type/iap/id/2001</uri>
  </operation-result>
</nam-response>
```

List Thresholds

Method	GET
URI	/nbi/nbi-threshold
Description	List all thresholds.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), internalError(-1), badParameter(-3), invalidID(-14)

Sample Request

This operation does not require a request body.

Sample Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <iapThreshold>
    <id>2001</id>
    <name>Response Time threshold</name>
    <severity>High</severity>
    <risingAction>1</risingAction>
    <fallingAction>1</fallingAction>
    <application>any</application>
    <server>
      <site>any</site>
      <host>any</host>
    </server>
    <iapMetrics>
      <risingAverageResponseTime>300</risingAverageResponseTime>
      <fallingAverageResponseTime>250</fallingAverageResponseTime>
      <risingMaxNetworkTime>100</risingMaxNetworkTime>
      <fallingMaxNetworkTime>80</fallingMaxNetworkTime>
    </iapMetrics>
  </iapThreshold>
</nam-response>
```

List Threshold Details

Method	GET
URI	/nbi/nbi-threshold/id/id
Description	List configuration details of a threshold.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), internalError(-1), badParameter(-3), invalidID(-14)

Sample Request

This operation does not require a request body.

Sample Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
    <operation-result>
        <status>0</status>
        <description>Successful</description>
    </operation-result>
    <iapThreshold>
        <id>2001</id>
        <name>Response Time threshold</name>
        <severity>High</severity>
        <risingAction>1</risingAction>
        <fallingAction>1</fallingAction>
        <application>any</application>
        <server>
            <site>any</site>
            <host>any</host>
        </server>
        <iapMetrics>
            <risingAverageResponseTime>300</risingAverageResponseTime>
            <fallingAverageResponseTime>250</fallingAverageResponseTime>
            <risingMaxNetworkTime>100</risingMaxNetworkTime>
            <fallingMaxNetworkTime>80</fallingMaxNetworkTime>
        </iapMetrics>
    </iapThreshold>
</nam-response>
```

Update Threshold

Method	PUT
URI	/nbi/nbi-threshold/type/`type`, where the supported threshold types are `host`, `convs`, `app`, `iap`, `dscp`, `rtp`, `voice`, and `nde`. Note that the threshold ID is specified in the XML request body.
Description	Update a threshold.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), internalError(-1), badParameter(-3), maxName(-6), siteIDError(-8), appIDError(-9), addressError(-10), metricError(-11), codecError(-12), severityIDError(-13), actionIDError(-14), dscpError(-19), invalidID(-20), typeError(-21), ifIndexError(-22), clntSrvrError(-23)

Sample Request

```
<iapThreshold>
  <id>2001</id>
  <name>Response Time threshold update</name>
  <severity>High</severity>
  <risingAction>2</risingAction>
  <fallingAction>2</fallingAction>
  <application>any</application>
  <server>
    <site>Any</site>
    <host>Any</host>
  </server>
  <iapMetrics>
    <risingAverageResponseTime>500</risingAverageResponseTime>
    <fallingAverageResponseTime>400</fallingAverageResponseTime>
    <risingMaxNetworkTime>200</risingMaxNetworkTime>
    <fallingMaxNetworkTime>100</fallingMaxNetworkTime>
  </iapMetrics>
</iapThreshold>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Delete Threshold

Method	DELETE
URI	/nbi/nbi-threshold/id/`id`
Description	Delete a threshold.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), internalError(500)
SECPA Status Code(s)	successful(0), internalError(-1), badParameter(-3)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Packet Capture

Packet capture allows network traffic to be recorded and saved for subsequent analysis and troubleshooting. This section provides an overview of packet capture concepts, but comprehensive coverage is beyond the scope of this document. For more detailed information on packet capture, see the "Capturing and Decoding Packets" section of SECPA User Guide.

The Packet Capture API supports various packet capture operations:

- Creating, editing, and deleting packet capture sessions.
- Starting, stopping, and clearing packet capture sessions.
- Configuring packet triggers, which can start a packet capture session when specified trigger conditions are met.
- Configuring hardware and software filters, which can prevent uninteresting packets from being captured.
- Listing, downloading, and deleting packet capture files.
- Creating, listing and deleting packet capture queries.
- Listing capture storage volumes (i.e., local disk(s), external storage).

Note that the REST API currently does not support decoding of packet captures; this must be done via the GUI.

All SECPA platforms support software filters. The SECPA with 2 x 10Gbps network interfaces also supports hardware filters. Hardware filtering is performed by dedicated filtering circuitry, whereas software filtering is performed by the SECPA software itself using the general-purpose CPU. Consequently, hardware filters are much faster and do not incur additional load on the SECPA software, but they are also limited in functionality and flexibility compared to software filters.

For a packet to be captured, it must first pass through the hardware filters (which are applied to all capture sessions), and then the software filters (which are applied on a per-session basis). Thus, hardware filters can be used to implement coarse filtering that reduces the number of packets that must be processed in software. Software filters can then be used to provide finer control over the packets actually captured to memory or disk. However, using hardware and software filters together is not required; each type can be used in isolation if desired, and using hardware filters alone is recommended if they are sufficient for the task at hand.

Hardware filters are implemented in the network interface, but not all SECPA platforms use the same network interface cards, so the exact hardware filter capabilities vary from platform to platform. To determine the filtering capabilities on a given SECPA, it is easiest to visit the Capture Sessions GUI page and see what options can be configured.

On SECPA platforms that support hardware filters, a packet matching **any** filter is considered an overall match, and matching packets are always included in the capture. Up to 5 hardware filters can be defined.

Software filters are implemented purely in software, so all SECPA platforms share a common implementation. To be included in a capture, a packet must simply match against **any** software filter (OR behavior). Software filters can be defined to match against combinations of the following criteria:

- Network encapsulation
- Application protocol
- IP protocol (TCP, UDP, or SCTP)
- Source and/or destination IP address (v4 and v6, with subnet mask)
- Source and/or destination port number (including ranges)
- VLAN ID (including ranges)

Each software filter is associated with a specific capture session, so that capture session must be defined prior to creating the software filter.

All SECPA platforms support external storage via iSCSI and SAS.

API Overview

URL	Method	Description
Capture Sessions API		
/nbi/nbi-capture/session	POST	Create a capture session.
/nbi/nbi-capture/session	GET	List all capture sessions.
/nbi/nbi-capture/session/id/`id`	GET	List configuration details of a capture session.
/nbi/nbi-capture/session/id/`id`	PUT	Update a capture session.
/nbi/nbi-capture/session/id/`id`	DELETE	Delete a capture session.
/nbi/nbi-capture/state/session/`id`/mode/`mode`	POST	Set the capture state/mode of a capture session.
Software Filters API		
/nbi/nbi-capture/swfilter	POST	Create a software filter.
/nbi/nbi-capture/swfilter	GET	List all software filters.
/nbi/nbi-capture/swfilter/id/`id`	PUT	Update a software filter.

URL	Method	Description
/nbi/nbi-capture/swfilter/id/id	DELETE	Delete a software filter.
Hardware Filters API		
/nbi/nbi-capture/hwfilter	POST	Create a hardware filter.
/nbi/nbi-capture/hwfilter	GET	List all hardware filters.
/nbi/nbi-capture/hwfilter/id/id	PUT	Update a hardware filter.
/nbi/nbi-capture/hwfilter/id/id	DELETE	Delete a hardware filter.
Capture Files API		
/nbi/nbi-capture/files	GET	List all capture files (with filename, size, and location).
/nbi/nbi-capture/capfiles/filename/filename	DELETE	Delete a capture file.
/capture/packets.php?captureFilename=filename	GET	Download a capture file.
Capture Queries API		
/nbi/nbi-capture/query	POST	Create and queue a capture query.
/nbi/nbi-capture/query[?[id=id1,id2,…][status=s1,s2,…][last=n]]	GET	List one or more capture queries. (Default: last 10 queries)
/nbi/nbi-capture/query?id=x[,y,…]	DELETE	Cancel one or more capture queries (which remain in the history for listing).
Capture Storage Volumes API		
/nbi/nbi-capture/storage	GET	List all capture storage volumes.

Capture Sessions

Create Capture Session

Method	POST
URI	/nbi/nbi-capture/session
Description	Create a capture session.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), resourceError(-2),maxSession(-3), maxFilters(-4), accessError(-5), badParameter(-6), internalError(-7), noSuchResource(-8), session name already exists(-9), capture to file already exists(-10)

Sample Request

Refer to Capture Session Elements for additional details on specific elements below.

Create a capture session backed by a 30MB memory buffer:

```
<capture>
  <session>
    <name>buffer_session</name>
    <trafficSource>1</trafficSource>
    <dataPorts>
      <dataPort>DATA PORT 1</dataPort>
      <dataPort>DATA PORT 2</dataPort>
    </dataPorts>
    <sliceSize>0</sliceSize>
    <state>0</state>
    <buffer>
      <bufferSize>30</bufferSize>
      <wrapMode>0</wrapMode>
    </buffer>
  </session>
</capture>
```

Create a capture session backed by 5 x 10MB files, rotate through the files, and only capture the first 500 bytes of each packet:

```
<capture>
  <session>
    <name>file_session</name>
    <trafficSource>1</trafficSource>
    <dataPorts>
      <dataPort>DATA PORT 1</dataPort>
      <dataPort>DATA PORT 2</dataPort>
    </dataPorts>
    <sliceSize>500</sliceSize>
    <state>0</state>
    <file>
      <rotateFiles>1</rotateFiles>
      <numFiles>5</numFiles>
      <fileSize>10</fileSize>
    </file>
  </session>
</capture>
```

Create a capture session backed by enough 2GB files to capture approximately 5Gbps of traffic for 10 minutes (600 seconds), make it uninterruptible by any other capture sessions, and start it automatically:

```
<capture>
  <session>
    <name>file_session</name>
    <trafficSource>1</trafficSource>
    <dataPorts>
      <dataPort>DATA PORT 1</dataPort>
      <dataPort>DATA PORT 2</dataPort>
    </dataPorts>
    <file>
      <timeDuration>600</timeDuration>
      <dataRate>5000</dataRate>
      <fileSize>2000</fileSize>
    </file>
    <uninterruptible>1</uninterruptible>
    <autoStart>1</autoStart>
  </session>
</capture>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
    <uri>/nbi/nbi-capture/session/id/2</uri>
  </operation-result>
</nam-response>
```

Capture Session Elements

Element	Description / Valid Values	Required
name	The name of the capture session.	✓
description	The description of the capture session.	
trafficSource	`1` (Data Ports), `2` (ERSPAN)	✓
dataPorts	A string of the form "DATA PORT **n**", where **n** is a digit between 1 and 4. The valid range is platform-dependent, and is shown in the SECPA GUI when creating a new capture session. This element is only used if `trafficSource` is `1`. If `trafficSource` is `1` but this element is omitted, then packets from all data ports will be captured.	
sliceSize	The packet offset at which the SECPA should slice off the remainder, in bytes. To capture the entire packet, either omit this element or specify a value of `0`.	

Element	Description / Valid Values	Required
state	Populated in the response. Either omit this element or use a dummy value of `0` in requests. `1` (Running), `2` (Cleared), `3` (Stopped on Error), `4` (Paused), `5` (Full), `6` (Saving File)	
buffer	Specifies that the capture session should store packet data in a memory buffer. It contains sub-elements, listed at Capture Session Buffer Elements. Either this or `file` is required to create a capture session.	✓
file	Specifies that the capture session should store packet data to files on disk. It contains sub-elements, listed at Capture Session Buffer Elements. Either this or `buffer` is required to create a capture session.	✓
filters	The software filters applied to the capture session. It contains sub-elements, listed at Capture Session Filter Elements. Note that this is status information only. Software filters cannot be created, updated, or associated with capture sessions via the Capture Sessions API — use the Software Filters API instead.	
timeTrigger	Specifies a time to start a scheduled capture session and how long to run it. It contains sub-elements, listed at Capture Session Time Trigger Elements. Not to be used with `uninterruptible`.	
uninterruptible	If set to `1`, the capture session will have the highest priority, meaning that any scheduled or alarm-triggered capture sessions will be blocked and cannot stop this session. Not to be used with `timeTrigger`.	
autoStart	Automatically starts the capture session upon creation.	
status	Provides details about the capture session status. It contains sub-elements, listed at Capture Session Status Elements.	

Capture Session Buffer Elements

Element	Description / Valid Values	Required
bufferSize	The size of the memory buffer, in MB. The max varies per platform and can be seen in the SECPA web interface on the Capture Sessions page.	✓
wrapMode	`0` (Stop capture when full), `1` (Wrap buffer when full)	✓

Capture Session File Elements

Element	Description / Valid Values	Required
rotateFiles	`0` (Stop capture when full), `1` (Rotate files when full)	
numFiles	The number of capture files for the capture session. Not to be used with `timeDuration` or `dataRate`.	✓

Element	Description / Valid Values	Required
timeDuration	The amount of time, in seconds, which should be captured by the session (approximately). This is used in combination with `dataRate` below. This, `dataRate`, and `fileSize` are used to calculate the number of files needed to meet the duration. Not to be used with `numFiles`.	
dataRate	The expected average data rate, in Mbps, while capturing. Use this with `timeDuration` (above) and `fileSize`. Not to be used with `numFiles`.	
fileSize	The approximate size of each file, in MB.	✓
fileLocation	Populated in the response. Useful for constructing a file decode or download URL.	
diskProtocol	Populated in the response. Useful for constructing a file decode or download URL.	
storageId	Specifies the storage device on which the capture files should be stored. The default is the local disk.	

Capture Session Filter Elements

Element	Description	Required
filterId	The ID of the software filter applied to the capture session. Up to 12 `filterId` elements can appear. Note that this is status information only. Software filters cannot be created, updated, or associated with capture sessions via the Capture Sessions API — use the Software Filters API instead.	✓

Capture Session Time Trigger Elements

Element	Description	Required
triggerStartTime	The time at which the capture session will be started, in the format `yyyy mm dd hh:mm:ss z`.	✓
maxDuration	The maximum duration of the capture session, in seconds. The capture session may stop prematurely if the memory buffer or files fill up first. In the files case, `rotateFiles` can be used to prevent that.	✓

Capture Session Status Elements

Element	Description	Required
startTime	The Unix time at which the capture session was last started.	✓
lastModificationTime	The Unix time at which the capture session was created or its parameters were last modified.	✓
packetsCaptured	The number of packets captured by the session thus far.	✓

List Capture Sessions

Method	GET
URI	/nbi/nbi-capture/session
Description	List all capture sessions.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), resourceError(-2), internalError(-7)

Sample Request

This operation does not require a request body.

Sample Response

Refer to Capture Session Elements for additional details on specific elements below.

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <capture>
    <session>
      <id>1</id>
      <name>buffer_session</name>
      <description></description>
      <trafficSource>1</trafficSource>
      <dataPorts>
        <dataPort>DATA PORT 1</dataPort>
        <dataPort>DATA PORT 2</dataPort>
      </dataPorts>
      <sliceSize>500</sliceSize>
      <state>Stopped</state>
      <buffer>
        <bufferSize>30</bufferSize>
        <wrapMode>0</wrapMode>
      </buffer>
      <filters></filters>
      <status>
        <startTime>0</startTime>
        <packetsCaptured>0</packetsCaptured>
      </status>
    </session>
    <session>
      <id>2</id>
```

```
        <name>file_session</name>
        <description></description>
        <trafficSource>1</trafficSource>
        <dataPorts>
          <dataPort>DATA PORT 1</dataPort>
          <dataPort>DATA PORT 2</dataPort>
        </dataPorts>
        <sliceSize>500</sliceSize>
        <state>Stopped</state>
        <file>
          <rotateFiles>1</rotateFiles>
          <numFiles>5</numFiles>
          <fileSize>10</fileSize>
          <fileLocation></fileLocation>
        </file>
        <filters></filters>
        <status>
          <startTime>0</startTime>
          <packetsCaptured>0</packetsCaptured>
        </status>
      </session>
    </capture>
  </nam-response>
```

List Capture Session Details

Method	GET
URI	/nbi/nbi-capture/session/id/`id`
Description	List configuration details of a capture session.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), resourceError(-2), internalError(-7)

Sample Request

This operation does not require a request body.

Sample Response

Refer to Capture Session Elements for additional details on specific elements below.

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <capture>
    <session>
      <id>1</id>
      <name>buffer_session</name>
      <description></description>
      <trafficSource>1</trafficSource>
      <dataPorts>
        <dataPort>DATA PORT 1</dataPort>
        <dataPort>DATA PORT 2</dataPort>
      </dataPorts>
      <sliceSize>500</sliceSize>
      <state>Stopped</state>
      <buffer>
        <bufferSize>30</bufferSize>
        <wrapMode>0</wrapMode>
      </buffer>
      <filters></filters>
      <status>
        <startTime>0</startTime>
        <packetsCaptured>0</packetsCaptured>
      </status>
    </session>
  </capture>
</nam-response>
```

Update Capture Session

Method	PUT
URI	/nbi/nbi-capture/session/id/id
Description	Update a capture session.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), resourceError(-2), maxSession(-3), maxFilters(-4), accessError(-5), badParameter(-6), internalError(-7), noSuchResource(-8)

Sample Request

Refer to Capture Session Elements for additional details on specific elements below.

```
<capture>
  <session>
    <name>buffer_session_updated</name>
    <trafficSource>1</trafficSource>
    <dataPorts>
      <dataPort>DATA PORT 1</dataPort>
    </dataPorts>
    <sliceSize>500</sliceSize>
    <state>0</state>
    <buffer>
      <bufferSize>50</bufferSize>
      <wrapMode>1</wrapMode>
    </buffer>
  </session>
</capture>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Delete Capture Session

Method	DELETE
URI	/nbi/nbi-capture/session/id/`id`
Description	Delete a capture session. The software filters associated with the capture session will also be deleted.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7), noSuchResource(-8)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Set Capture Session State

Method	POST
URI	/nbi/nbi-capture/state/session/`id`/mode/`mode` `mode` is one of: `3` (Start), `4` (Stop), `8` (Clear)
Description	Set the capture state/mode of a capture session. Note that setting a capture session state is equivalent to clicking the corresponding "Start", "Stop", or "Clear" button on the Capture Session GUI page, so the same state transition rules apply. For example, if a session is set to "Start" and runs until the buffer is full, then the session state must be set to "Clear" before it can be set to "Start" again.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7)

Sample Request

This operation does not require a request body.

For the capture session with ID 1, posting to `/nbi/nbi-capture/state/session/1/mode/3` would be equivalent to pressing the GUI’s "Start" button, and posting to `/nbi/nbi-capture/state/session/1/mode/4` would be equivalent to pressing the "Stop" button.

Note that there is no `id` component in the URI path; that is, `/nbi/nbi-capture/state/session/id/1/mode/3` would result in an error.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Software Filters

Create Software Filter

Method	POST
URI	/nbi/nbi-capture/swfilter
Description	Create a software filter. Since a software filter is associated with a specific capture session, that capture session must be created before any of its software filters.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7)

Sample Request

Refer to Software Filter Elements for additional details on specific elements below.

```
<capture>
  <swFilter>
    <name>test</name>
    <sessionId>1</sessionId>
    <srcAddrMask>10.0.0.4/16</srcAddrMask>
    <dstAddrMask>10.0.0.8/16</dstAddrMask>
    <netEncap>1</netEncap>
    <vlans>11</vlans>
    <application>16777220</application>
    <direction>1</direction>
  </swFilter>
</capture>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
    <uri>/nbi/nbi-capture/swfilter/id/1</uri>
  </operation-result>
</nam-response>
```

Software Filter Elements

Element	Description / Valid Values
sessionId	The ID of the capture session with which the software filter is associated. This is returned as part of the XML response when creating a capture session.
srcAddrMask / dstAddrMask	An IP subnet address and mask (e.g. `1.2.3.4/16`).
srcPort / dstPort	A port number, used with port-based protocols like TCP, UDP, or SCTP.
netEncap	`170` (CAPWAP Data), `240` (ERSPAN), `40` (FabricPath), `90` (GRE), `130` (GTP), `100` (IP.IP4), `110` (IP.IP6), `120` (IPESP), `140` (L2TP Data), `80` (LISP Data), `190` (LWAP Data), `60` (MPLS), `70` (OTV), `160` (PPPoE), `50` (SegmentID), `220` (SGT), `30` (VNTAG), `20` (VxLAN)
vlans	A single VLAN ID (e.g., `1`), or a range of VLAN IDs (e.g., `1-4`).
protocol	`0` (Any), `6` (TCP), `17` (UDP), `132` (SCTP)
application	An apptag, which can be retrieved using the Applications API.
direction	`0` (One direction), `1` (Both directions)

List Software Filters

Method	GET
URI	/nbi/nbi-capture/swfilter
Description	List all software filters.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7)

Sample Request

This operation does not require a request body.

Sample Response

Refer to Software Filter Elements for additional details on specific elements below.

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <capture>
    <swFilter>
      <name>test</name>
      <filterId>1</filterId>
      <sessionId>1</sessionId>
      <srcAddrMask>10.0.0.4/16</srcAddrMask>
      <dstAddrMask>10.0.0.8/16</dstAddrMask>
      <vlans>11</vlans>
      <application>16777220</application>
      <direction>1</direction>
    </swFilter>
  </capture>
</nam-response>
```

Update Software Filters

Method	PUT
URI	/nbi/nbi-capture/swfilter/id/id
Description	Update a software filter.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7), noSuchResource(-8)

Sample Request

Refer to Software Filter Elements for additional details on specific elements below.

```
<capture>
  <swFilter>
    <name>test_update</name>
    <sessionId>1</sessionId>
    <srcPort>45</srcPort>
    <protocol>tcp</protocol>
  </swFilter>
</capture>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Delete Software Filter

Method	DELETE
URI	/nbi/nbi-capture/swfilter/id/`id`
Description	Delete a software filter.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7), noSuchResource(-8)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Hardware Filters

Create Hardware Filter

Method	POST
URI	/nbi/nbi-capture/hwfilter
Description	Create a hardware filter.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7)

Sample Request

Refer to Hardware Filter Elements for additional details on specific elements below.

```
<capture>
  <hwFilter>
    <name>hwfilter_test</name>
    <filterType>4</filterType> <!-- IP and TCP/UDP -->
    <ipVersion>4</ipVersion>
    <srcAddrMask>10.0.0.1/32</srcAddrMask>
    <dstAddrMask>10.0.0.5/32</dstAddrMask>
    <srcPort>80</srcPort>
    <protocol>6</protocol>
  </hwFilter>
</capture>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
    <uri>/nbi/nbi-capture/hwfilter/id/1</uri>
  </operation-result>
</nam-response>
```

Hardware Filter Elements

Element	Description / Valid Values
filterType	Specifies the type of hardware filter being defined. `0` (VLAN), `1` (VLAN and IP), `3` (IP), `4` (IP and TCP/UDP), `5` (IP and Payload Data), `6` (Payload Data)
protocol	`0` (Any), `1` (ICMP), `2` (IGMP), `6` (TCP), `17` (UDP), `47` (GRE), `94` (IP-in-IP), `115` (L2TP)

List Hardware Filters

Method	GET
URI	/nbi/nbi-capture/hwfilter
Description	List all hardware filters.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7)

Sample Request

This operation does not require a request body.

Sample Response

Refer to Hardware Filter Elements for additional details on specific elements below.

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <capture>
    <hwFilter>
      <name>hwfilter_test</name>
      <filterId>1</filterId>
      <filterType>4</filterType>
      <ipVersion>4</ipVersion>
      <srcAddrMask>10.0.0.1/32</srcAddrMask>
      <dstAddrMask>10.0.0.5/32</dstAddrMask>
      <srcPort>80</srcPort>
      <protocol>6</protocol>
    </hwFilter>
  </capture>
</nam-response>
```

Update Hardware Filters

Method	PUT
URI	/nbi/nbi-capture/hwfilter/id/id
Description	Update a hardware filter.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7), noSuchResource(-8)

Sample Request

Refer to Hardware Filter Elements for additional details on specific elements below.

```
<capture>
  <hwFilter>
    <name>hwfilter_test_disabled</name>
    <filterType>4</filterType>
    <ipVersion>4</ipVersion>
    <srcAddrMask>10.0.0.1/32</srcAddrMask>
    <dstAddrMask>10.0.0.5/32</dstAddrMask>
    <srcPort>80</srcPort>
    <protocol>6</protocol>
    <filterStatus>disable</filterStatus>
  </hwFilter>
</capture>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Delete Hardware Filter

Method	DELETE
URI	/nbi/nbi-capture/hwfilter/id/id
Description	Delete a hardware filter.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7), noSuchResource(-8)

Sample Request

This operation does not require a request body.

Sample Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Capture Files

List Capture Files

Method	GET
URI	/nbi/nbi-capture/files
Description	List all capture files (with filename, size, and location).
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7), noSuchResource(-8)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <captureFiles>
    <totalDiskStorage>470189905920</totalDiskStorage>
    <availDiskStorage>303317385216</availDiskStorage>
    <downloadFileUri>/capture/packets.php?captureSessionId=0&amp;
captureFilename=</downloadFileUri>
    <files>
      <file>
        <name>test2_1.pcap</name>
        <size>10485331</size>
        <date>1387240860</date>
        <location>/storage/capture</location>
        <state>5</state>
      </file>
      <file>
        <name>test1_1.pcap</name>
        <size>52428676</size>
        <date>1387240810</date>
        <location>/storage/capture</location>
        <state>5</state>
      </file>
      <file>
        <name>test2_2.pcap</name>
        <size>10485743</size>
        <date>1387240861</date>
        <location>/storage/capture</location>
        <state>5</state>
      </file>
    </files>
  </captureFiles>
</nam-response>
```

Delete Capture File

Method	DELETE
URI	/nbi/nbi-capture/capfiles/filename/`filename`
Description	Delete a capture file.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400)
SECPA Status Code(s)	successful(0), badParameter(-4)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Download Capture File

Method	GET
URI	/capture/packets.php?captureFilename=`filename`
Description	Download a capture file.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), resourceError(-2), badParameter(-6), internalError(-7), noSuchResource(-8)

Sample Request

This operation does not require a request body.

Sample Response

This operation does not return an XML response, just the raw packet file data.

Capture Queries

Capture queries allow users to search for packets in capture session files, and in turn, generate pcap files containing just the packets matching the search criteria. The search criteria must include at least one of the following:

- A filter string in display filter syntax.
- A time range start time.
- A time range end time.

If the capture session is running when a query is created, the query cannot inspect the capture file that is currently being written. The maximum capture file size, 2GB, is recommended for optimal capture and query performance.

Create Capture Query

Method	POST
URI	/nbi/nbi-capture/query
Description	Create a capture query.
HTTP Normal Response Code	successful(200)
HTTP Error Response Codes	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Codes	successful(0), resourceError(-1), invalidNbiParams(-2), invalidIdListCount(-3), invalidStatus(-4), invalidLastN(-5)
Capture Query Status Codes	queued(1), inProgress(2), complete(4), noResults(8), canceled(16), invalidQueryParams(32), internalQueryError(64), doesNotExist(128), deleted(256), archiving(512)

Sample Request

Refer to Capture Query Elements for additional details on specific elements below.

Create a simple query using just the required elements:

```
<capture>
    <query>
        <name>lldp_102_bytes</name>
        <filter>lldp and frame.len == 102</filter>
        <sessionId>1</sessionId>
    </query>
</capture>
```

Create a complex query using all of the optional elements:

```
<capture>
    <query>
        <name>http_from_a_server</name>
        <filter>http and ip.src == 50.6.10.157</filter>
        <filterFromTime>1455750000</filterFromTime>
        <filterToTime>1455760000</filterToTime>
        <sessionId>1</sessionId>
        <maxFileSize>100</maxFileSize>
        <storageId>1</storageId>
    </query>
</capture>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
    <operation-result>
        <status>0</status>
        <description>Success</description>
    </operation-result>
    <capture>
        <query>
            <id>55</id>
            <status>1</status>
            <message>Capture query queued successfully.</message>
            <user>admin</user>
            <name>udp</name>
            <filter>udp</filter>
            <filterFromTime>0</filterFromTime>
            <filterToTime>0</filterToTime>
            <sessionId>1</sessionId>
            <maxFileSize>2000</maxFileSize>
            <storageId>1</storageId>
            <queueTime>1460484361</queueTime>
            <runTime>0</runTime>
            <completeTime>0</completeTime>
            <progress>0</progress>
            <archive></archive>
            <files></files>
            <numPackets>0</numPackets>
        </query>
    </capture>
</nam-response>
```

Capture Query Elements

Element	Description / Valid Values	Required
id	The ID of the capture query.	
status	The possible statuses are listed under Capture Query Status Codes.	
message	A message that explains the status and any specific details. (Max: 127 chars)	
user	The SECPA web user or TACACS+ user who created the query.	
name	A meaningful name for the query. It can only contain letters, numbers, "-" and "_". (Max: 63 chars)	✓
filter	The filter string in the same display filter syntax as the SECPA decoder. (Max: 255 chars)	✓
sessionId	The ID of the capture session to query. Refer to Capture Sessions API for getting session info.	✓

Element	Description / Valid Values	Required
filterFromTime	The start of the time range in which the query will look for packets, as a Unix time (e.g., 1455750000). If omitted, the start time is the timestamp of the oldest packet in the specified capture session.	
filterToTime	The end of the time range in which the query will look for packets, as a Unix time. If omitted, the end time is the time at which the query is executed (if the capture session is running), or the timestamp of the last packet (if the capture session is stopped).	
maxFileSize	The maximum size of each pcap file (1-2000 MB). In other words, as the query is generating the pcap result, it will be split into pcap files of `maxFileSize` MB each (except for the last file, which may be smaller).	
storageId	The ID of the capture storage device on which the query results will be stored. If omitted, the local storage is used. Refer to Capture Storage Volumes API for getting storage info.	
queueTime	The time at which the query was queued.	
runTime	The time at which the query started running.	
completeTime	The time at which the query finished.	
resultsFromTime	The start of the time range of the query results, as a Unix time.	
resultsToTime	The end of the time range of the query results, as a Unix time.	
progress	The progress of the query, in percentage.	
archive	If the query has any results, all of the pcap files will be put into a compressed archive (.tgz). If so, this element contains the sub-elements listed at Capture Query File Elements.	
files	If the query has any results, this will contain a `file` sub-element result file (.pcap). Each `file` sub-element has its own sub-elements, listed at Capture Query File Elements. They can be downloaded using the Download Capture File endpoint.	
numPackets	The number of packets matching the query parameters.	

Capture Query File Elements

Element	Description / Valid Values	Required
uri	The full URI of the file.	✓
size	The size of the file in bytes.	✓

List Capture Queries

Method	GET

URI	/nbi/nbi-capture/query
Description	List one or more capture queries.
Parameters	All of the parameters are optional. The default behavior is to return the last 10 queries. `id=x,y,⋯` — List queries with IDs X, Y, etc. The maximum list length is 100 IDs. Ranges are also supported (e.g., `?id=1-5`) as long as the range is not larger than 100 IDs. Note that this API does not support listing by query names. `last=N` — List the last N queries. The maximum value is 100. If omitted, the default is 10. Use of `id` overrides `last`. `status=running,⋯` — Filter the query list for queries that match one of the given statuses. The list can contain status code values and/or names. See Capture Query Status Codes for valid values. Valid status names are `queued`, `running`, `complete`, `noResults`, `canceled`, `failed`. The `running` status includes the status codes for `inProgress` & `archiving`. The `failed` status encompasses all error statuses.
HTTP Normal Response Code	successful(200)
HTTP Error Response Codes	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Codes	successful(0), resourceError(-1), invalidNbiParams(-2)
Capture Query Status Codes	queued(1), inProgress(2), complete(4), noResults(8), canceled(16), invalidQueryParams(32), internalQueryError(64), doesNotExist(128), deleted(256), archiving(512)

Sample Requests and Responses

Refer to Capture Query Elements for additional details on specific elements below.

`/nbi/nbi-capture/query?last=3&status=noResults` (of the last 3 queries, list the ones with no results):

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
    <operation-result>
        <status>0</status>
        <description>Success</description>
    </operation-result>
    <capture>
        <query>
            <id>3</id>
            <status>8</status>
            <message>Complete with no results.</message>
            <user>admin</user>
            <name>1_1_1_1</name>
            <filter>ip.src == 1.1.1.1</filter>
            <filterFromTime>0</filterFromTime>
            <filterToTime>0</filterToTime>
            <sessionId>1</sessionId>
            <maxFileSize>2000</maxFileSize>
            <storageId>1</storageId>
            <queueTime>1465503562</queueTime>
            <runTime>1465503563</runTime>
            <completeTime>1465503565</completeTime>
            <resultsFromTime>1465503347</resultsFromTime>
            <resultsToTime>1465503362</resultsToTime>
            <progress>100</progress>
            <archive/>
            <files></files>
            <numPackets>0</numPackets>
        </query>
    </capture>
</nam-response>
```

`/nbi/nbi-capture/query?id=1,3` (list the queries with IDs 1 and 3):

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
    <operation-result>
        <status>0</status>
        <description>Success</description>
    </operation-result>
    <capture>
        <query>
            <id>1</id>
            <status>4</status>
            <message>Complete with results.</message>
            <user>admin</user>
            <name>udp</name>
            <filter>udp</filter>
            <filterFromTime>0</filterFromTime>
            <filterToTime>0</filterToTime>
            <sessionId>1</sessionId>
            <maxFileSize>2000</maxFileSize>
            <storageId>1</storageId>
            <queueTime>1465503390</queueTime>
            <runTime>1465503391</runTime>
            <completeTime>1465503399</completeTime>
            <resultsFromTime>1465503347</resultsFromTime>
            <resultsToTime>1465503362</resultsToTime>
            <progress>100</progress>
            <archive>
                <uri>/capture/packets.php?captureFilename=query1_udp.tgz</uri>
                <size>20432311</size>
            </archive>
            <files>
                <file>
                    <uri>/capture/packets.php?captureFilename=query1_udp_1.pcap</uri>
                    <size>95101165</size>
                </file>
            </files>
            <numPackets>253256</numPackets>
        </query>
        <query>
            <id>3</id>
            <status>8</status>
            <message>Complete with no results.</message>
            <user>admin</user>
            <name>1_1_1_1</name>
            <filter>ip.src == 1.1.1.1</filter>
            <filterFromTime>0</filterFromTime>
            <filterToTime>0</filterToTime>
            <sessionId>1</sessionId>
            <maxFileSize>2000</maxFileSize>
            <storageId>1</storageId>
            <queueTime>1465503562</queueTime>
            <runTime>1465503563</runTime>
            <completeTime>1465503565</completeTime>
            <resultsFromTime>1465503347</resultsFromTime>
            <resultsToTime>1465503362</resultsToTime>
            <progress>100</progress>
            <archive/>
            <files></files>
            <numPackets>0</numPackets>
        </query>
    </capture>
</nam-response>
```

Cancel Capture Queries

Method	PUT

URI	/nbi/nbi-capture/query
Description	Cancel a capture query.
HTTP Normal Response Code	successful(200)
HTTP Error Response Codes	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Codes	successful(0), resourceError(-1), invalidNbiParams(-2)
Capture Query Status Codes	queued(1), inProgress(2), complete(4), noResults(8), canceled(16), invalidQueryParams(32), internalQueryError(64), doesNotExist(128), deleted(256), archiving(512)

Sample Request

Refer to Capture Query Update Elements for additional details on specific elements below.

Cancel the query with ID 52:

```
<capture>
    <queryUpdate>
        <id>52</id>
        <status>16</status>
    </queryUpdate>
</capture>
```

Sample Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
    <operation-result>
        <status>0</status>
        <description>Success</description>

    </operation-result>
    <capture>
        <query>
            <id>52</id>
            <status>16</status>
            <message>Capture query canceled while in progress.</message>
            <user>admin</user>
            <name>cancel2</name>
            <filter>udp</filter>
            <filterFromTime>1459205943</filterFromTime>
            <filterToTime>1460441702</filterToTime>
            <sessionId>2</sessionId>
            <maxFileSize>2000</maxFileSize>
            <storageId>1</storageId>
            <queueTime>1460441701</queueTime>
            <runTime>1460441702</runTime>
            <completeTime>1460441747</completeTime>
            <progress>10</progress>
            <archive></archive>
            <files></files>
            <numPackets>47068</numPackets>
        </query>
    </capture>
</nam-response>
```

Capture Query Update Elements

Element	Description / Valid Values	Required
id	The ID of the capture query.	✓
status	The only currently supported update status is `16` (canceled).	✓

Delete Capture Query Files

Method	DELETE
URI	/nbi/nbi-capture/query
Description	Delete all of the output files associated with a capture query (.pcap and .tgz).
Parameters	`id=x,y,…` — List queries with IDs X, Y, etc. The maximum list length is 100 IDs. Ranges are also supported (e.g., `id=1-5`), as long as the range is not larger than 100 IDs.

HTTP Normal Response Code	successful(200)
HTTP Error Response Codes	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Codes	successful(0), resourceError(-1), invalidNbiParams(-2)
Capture Query Status Codes	queued(1), inProgress(2), complete(4), noResults(8), canceled(16), invalidQueryParams(32), internalQueryError(64), doesNotExist(128), deleted(256), archiving(512)

Sample Requests and Responses

Refer to Capture Query Elements for additional details on specific elements below.

DELETE `/nbi/nbi-capture/query?id=56,57` (delete files from queries 56 and 57):

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
    <operation-result>
        <status>0</status>
        <description>Success</description>
    </operation-result>
    <capture></capture>
</nam-response>
```

After the DELETE operation, listing the queries will result in "Files Deleted" messages, and the files are gone.

GET `/nbi/nbi-capture/query?id=56,57` (list queries 56 and 57):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
    <operation-result>
        <status>0</status>
        <description>Success</description>
    </operation-result>
    <capture>
        <query>
            <id>56</id>
            <status>256</status>
            <message>Files deleted</message>
            <user>admin</user>
            <name>udp</name>
            <filter>udp</filter>
            <filterFromTime>1459900017</filterFromTime>
            <filterToTime>1459900021</filterToTime>
            <sessionId>1</sessionId>
            <maxFileSize>2000</maxFileSize>
            <storageId>1</storageId>
            <queueTime>1460489154</queueTime>
            <runTime>1460489154</runTime>
            <completeTime>1460489156</completeTime>
            <progress>100</progress>
            <archive></archive>
            <files></files>
            <numPackets>48348</numPackets>
        </query>
        <query>
            <id>57</id>
            <status>256</status>
            <message>Files deleted</message>
            <user>admin</user>
            <name>udp</name>
            <filter>udp</filter>
            <filterFromTime>1459900017</filterFromTime>
            <filterToTime>1459900021</filterToTime>
            <sessionId>1</sessionId>
            <maxFileSize>10</maxFileSize>
            <storageId>1</storageId>
            <queueTime>1460489232</queueTime>
            <runTime>1460489232</runTime>
            <completeTime>1460489234</completeTime>
            <progress>100</progress>
            <archive></archive>
            <files></files>
            <numPackets>48348</numPackets>
        </query>
    </capture>
</nam-response>
```

Capture Storage Volumes

List Capture Storage Volumes

Method	GET
URI	/nbi/nbi-capture/storage
Description	List all capture storage volumes.
HTTP Normal Response Code	successful(200)
HTTP Error Response Codes	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Codes (error codes are negative)	successful(0), resourceError(-1), invalidParams(-2)

Sample Request

The operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
    <operation-result>
        <status>0</status>
        <description>Successful.</description>
    </operation-result>
    <capture>
        <storage>
            <id>1</id>
            <label>Local Disk</label>
            <status>4</status>
            <protocol>SAS</protocol>
            <model>Cisco UCSC-MRAID12G</model>
            <capacity>20978985861120</capacity>
            <capacity_readable>19.08TB</capacity_readable>
            <available>20004029333504</available>
            <available_readable>18.19TB</available_readable>
            <ip></ip>
            <iqn></iqn>
            <lun>0</lun>
            <localDisk>1</localDisk>
        </storage>
    </capture>
</nam-response>
```

Capture Storage Elements

Element	Description / Valid Values
id	The ID of the storage volume.
label	The user defined label for the storage volume.
status	Status codes are: unformatted(1), unavailable(2), userUnmounted(3), ready(4), inUse(5)
protocol	The storage protocol used by the volume (FCOE, ISCSI, SAS, SCSI).
model	The vendor and storage model associated with the volume.
capacity	The total capacity of the volume, in bytes.
capacityReadable	The total capacity of the volume in a human readable format (e.g., 38.01TB).
available	The available space in the volume, in bytes.
availableReadable	The available space in the volume in a human readable format.
ip	For an iSCSI volume, the target IP address.
iqn	For an iSCSI volume, the iSCSI Qualified Name.
lun	The logical unit number. Note that a single physical storage array can have multiple logical volumes; each volume gets a unique number within that array.
localDisk	Boolean integer indicating if the volume is the SECPA's local disk.

System Info

The System Info API provides various details about the SECPA hardware and software, such as the SECPA platform/model identifier, software version, and CPU and disk usage statistics.

API Overview

URL	Method	Description
/nbi/nbi-system	GET	List system information.

List System Info

Method	GET
URI	/nbi/nbi-system
Description	List system information.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), failed(-1)

Sample Request

This operation does not require a request body.

Sample Response

Note: `/storage` contains packet capture files, and `/storage1` contains CDB database files The `oldestDataTime` and `newestDataTime` elements represent the Unix times of the oldest and newest data recorded in the corresponding CDB file.

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <systemInfo>
    <hardware>
      <name>Cisco Security Packet Analyzer</name>
      <model>SEC-PA-2400-K9</model>
      <serial>FCH2013V0JB</serial>
    </hardware>
    <software>
      <appVersion>6.2(2)</appVersion>
    </software>
    <systemTime>
      <uptime>16392272</uptime>
    </systemTime>
    <cpuInfo>
      <cpu>
        <id>999</id>
        <usrpct>2</usrpct>
        <nicepct>0</nicepct>
        <syspct>2.36</syspct>
        <iowaitpct>0.06</iowaitpct>
        <irqpct>0</irqpct>
        <softpct>0</softpct>
        <stealpct>0</stealpct>
        <guestpct>0</guestpct>
        <idlepct>95.58</idlepct>
      </cpu>
      <cpu>
        <id>0</id>
        <usrpct>1.44</usrpct>
        <nicepct>0</nicepct>
        <syspct>5.21</syspct>
        <iowaitpct>0.01</iowaitpct>
        <irqpct>0</irqpct>
        <softpct>0.15</softpct>
        <stealpct>0</stealpct>
        <guestpct>0</guestpct>
        <idlepct>93.19</idlepct>
      </cpu>
      <cpu>
        <id>1</id>
        <usrpct>1.49</usrpct>
        <nicepct>0</nicepct>
        <syspct>5.09</syspct>
        <iowaitpct>0.01</iowaitpct>
        <irqpct>0</irqpct>
        <softpct>0</softpct>
        <stealpct>0</stealpct>
        <guestpct>0</guestpct>
        <idlepct>93.42</idlepct>
      </cpu>
      <cpu>
        <id>2</id>
        <usrpct>0</usrpct>
        <nicepct>0</nicepct>
        <syspct>0</syspct>
        <iowaitpct>0</iowaitpct>
        <irqpct>0</irqpct>
        <softpct>0</softpct>
        <stealpct>0</stealpct>
        <guestpct>0</guestpct>
        <idlepct>100</idlepct>
      </cpu>
      <cpu>
        <id>3</id>
        <usrpct>0</usrpct>
        <nicepct>0</nicepct>
        <syspct>0</syspct>
        <iowaitpct>0</iowaitpct>
        <irqpct>0</irqpct>
        <softpct>0</softpct>
        <stealpct>0</stealpct>
        <guestpct>0</guestpct>
        <idlepct>100</idlepct>
      </cpu>
      <cpu>
        <id>4</id>
        <usrpct>0</usrpct>
        <nicepct>0</nicepct>
        <syspct>0</syspct>
        <iowaitpct>0</iowaitpct>
        <irqpct>0</irqpct>
        <softpct>0</softpct>
        <stealpct>0</stealpct>
        <guestpct>0</guestpct>
        <idlepct>100</idlepct>
      </cpu>
      <cpu>
      <id>5</id>
      <usrpct>0</usrpct>
      <nicepct>0</nicepct>
      <syspct>0</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>100</idlepct>
    </cpu>
    <cpu>
      <id>6</id>
      <usrpct>0</usrpct>
      <nicepct>0</nicepct>
      <syspct>0</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>100</idlepct>
    </cpu>
    <cpu>
      <id>7</id>
      <usrpct>0</usrpct>
      <nicepct>0</nicepct>
      <syspct>0</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>100</idlepct>
    </cpu>
    <cpu>
      <id>8</id>
      <usrpct>0</usrpct>
      <nicepct>0</nicepct>
      <syspct>0</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>100</idlepct>
    </cpu>
    <cpu>
      <id>9</id>
      <usrpct>0</usrpct>
      <nicepct>0</nicepct>
```

```
      <syspct>0</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>100</idlepct>
    </cpu>
    <cpu>
      <id>10</id>
      <usrpct>0</usrpct>
      <nicepct>0</nicepct>
      <syspct>0</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>100</idlepct>
    </cpu>
    <cpu>
      <id>11</id>
      <usrpct>0</usrpct>
      <nicepct>0</nicepct>
      <syspct>0</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>100</idlepct>
    </cpu>
    <cpu>
      <id>12</id>
      <usrpct>0</usrpct>
      <nicepct>0</nicepct>
      <syspct>0</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>100</idlepct>
    </cpu>
    <cpu>
      <id>13</id>
      <usrpct>0.02</usrpct>
      <nicepct>0</nicepct>
      <syspct>0.07</syspct>
      <iowaitpct>0.09</iowaitpct>
      <irqpct>0</irqpct>
```

```
        <softpct>0</softpct>
        <stealpct>0</stealpct>
        <guestpct>0</guestpct>
        <idlepct>99.82</idlepct>
      </cpu>
      <cpu>
        <id>14</id>
        <usrpct>0.02</usrpct>
        <nicepct>0</nicepct>
        <syspct>0.07</syspct>
        <iowaitpct>0.09</iowaitpct>
        <irqpct>0</irqpct>
        <softpct>0</softpct>
        <stealpct>0</stealpct>
        <guestpct>0</guestpct>
        <idlepct>99.82</idlepct>
      </cpu>
      <cpu>
        <id>15</id>
        <usrpct>0.02</usrpct>
        <nicepct>0</nicepct>
        <syspct>0.07</syspct>
        <iowaitpct>0.09</iowaitpct>
        <irqpct>0</irqpct>
        <softpct>0</softpct>
        <stealpct>0</stealpct>
        <guestpct>0</guestpct>
        <idlepct>99.82</idlepct>
      </cpu>
      <cpu>
        <id>16</id>
        <usrpct>0.02</usrpct>
        <nicepct>0</nicepct>
        <syspct>0.07</syspct>
        <iowaitpct>0.09</iowaitpct>
        <irqpct>0</irqpct>
        <softpct>0</softpct>
        <stealpct>0</stealpct>
        <guestpct>0</guestpct>
        <idlepct>99.82</idlepct>
      </cpu>
      <cpu>
        <id>17</id>
        <usrpct>0.01</usrpct>
        <nicepct>0</nicepct>
        <syspct>0.07</syspct>
        <iowaitpct>0.09</iowaitpct>
        <irqpct>0</irqpct>
        <softpct>0</softpct>
        <stealpct>0</stealpct>
        <guestpct>0</guestpct>
      <idlepct>99.83</idlepct>
    </cpu>
    <cpu>
      <id>18</id>
      <usrpct>0.01</usrpct>
      <nicepct>0</nicepct>
      <syspct>0.07</syspct>
      <iowaitpct>0.09</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>99.83</idlepct>
    </cpu>
    <cpu>
      <id>19</id>
      <usrpct>0.01</usrpct>
      <nicepct>0</nicepct>
      <syspct>0.07</syspct>
      <iowaitpct>0.09</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>99.82</idlepct>
    </cpu>
    <cpu>
      <id>20</id>
      <usrpct>0.01</usrpct>
      <nicepct>0</nicepct>
      <syspct>0.07</syspct>
      <iowaitpct>0.09</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>99.83</idlepct>
    </cpu>
    <cpu>
      <id>21</id>
      <usrpct>0.01</usrpct>
      <nicepct>0</nicepct>
      <syspct>0.07</syspct>
      <iowaitpct>0.09</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>99.82</idlepct>
    </cpu>
    <cpu>
```

```
      <id>22</id>
      <usrpct>0.01</usrpct>
      <nicepct>0</nicepct>
      <syspct>0.07</syspct>
      <iowaitpct>0.09</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>99.82</idlepct>
    </cpu>
    <cpu>
      <id>23</id>
      <usrpct>0.96</usrpct>
      <nicepct>0</nicepct>
      <syspct>4.77</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>94.27</idlepct>
    </cpu>
    <cpu>
      <id>24</id>
      <usrpct>3.36</usrpct>
      <nicepct>0</nicepct>
      <syspct>4.45</syspct>
      <iowaitpct>1.35</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>90.84</idlepct>
    </cpu>
    <cpu>
      <id>25</id>
      <usrpct>5.35</usrpct>
      <nicepct>0</nicepct>
      <syspct>23.98</syspct>
      <iowaitpct>0.02</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>70.64</idlepct>
    </cpu>
    <cpu>
      <id>26</id>
      <usrpct>4.95</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.95</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>91.1</idlepct>
    </cpu>
    <cpu>
      <id>27</id>
      <usrpct>4.91</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.99</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>91.1</idlepct>
    </cpu>
    <cpu>
      <id>28</id>
      <usrpct>4.98</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.96</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>91.06</idlepct>
    </cpu>
    <cpu>
      <id>29</id>
      <usrpct>5.13</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.98</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>90.89</idlepct>
    </cpu>
    <cpu>
      <id>30</id>
      <usrpct>5.15</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.57</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>91.28</idlepct>
    </cpu>
    <cpu>
      <id>31</id>
      <usrpct>4.88</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.73</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>91.39</idlepct>
    </cpu>
    <cpu>
      <id>32</id>
      <usrpct>4.99</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.67</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>91.34</idlepct>
    </cpu>
    <cpu>
      <id>33</id>
      <usrpct>4.89</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.68</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>91.43</idlepct>
    </cpu>
    <cpu>
      <id>34</id>
      <usrpct>4.88</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.74</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>91.38</idlepct>
    </cpu>
    <cpu>
      <id>35</id>
      <usrpct>4.96</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.75</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>91.28</idlepct>
    </cpu>
    <cpu>
      <id>36</id>
      <usrpct>4.77</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.69</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>91.53</idlepct>
    </cpu>
    <cpu>
      <id>37</id>
      <usrpct>4.86</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.73</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>91.41</idlepct>
    </cpu>
    <cpu>
      <id>38</id>
      <usrpct>5.19</usrpct>
      <nicepct>0</nicepct>
      <syspct>3.91</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>90.9</idlepct>
    </cpu>
    <cpu>
```

```
      <id>39</id>
      <usrpct>2.56</usrpct>
      <nicepct>0</nicepct>
      <syspct>0.9</syspct>
      <iowaitpct>0</iowaitpct>
      <irqpct>0</irqpct>
      <softpct>0</softpct>
      <stealpct>0</stealpct>
      <guestpct>0</guestpct>
      <idlepct>96.54</idlepct>
    </cpu>
  </cpuInfo>
  <memoryInfo>
    <total>132190944</total>
    <used>20745012</used>
    <free>68358084</free>
    <cached>42991368</cached>
    <buffer>96480</buffer>
  </memoryInfo>
  <diskInfo>
    <filesystem>
      <name>/</name>
      <size>104037172</size>
      <used>1512928</used>
      <free>97281032</free>
      <use>2%</use>
    </filesystem>
    <filesystem>
      <name>/tmp</name>
      <size>13219096</size>
      <used>100</used>
      <free>13218996</free>
      <use>1%</use>
    </filesystem>
    <filesystem>
      <name>/nvram</name>
      <size>1043900</size>
      <used>40012</used>
      <free>951276</free>
      <use>5%</use>
    </filesystem>
    <filesystem>
      <name>/storage1</name>
      <size>1831684696</size>
      <used>782420876</used>
      <free>956952324</free>
      <use>45%</use>
    </filesystem>
    <filesystem>
      <name>/storage</name>
      <size>32791854592</size>
```

```
          <used>16027887216</used>
          <free>16763967376</free>
          <use>49%</use>
        </filesystem>
        <filesystem>
          <name>/mnt/payload</name>
          <size>66095472</size>
          <used>0</used>
          <free>66095472</free>
          <use>0%</use>
        </filesystem>
      </diskInfo>
      <nicDrops>
        <nic>
          <absolute>0</absolute>
          <delta10sec>0</delta10sec>
        </nic>
      </nicDrops>
      <flowDrops>
        <flow>
          <name>art</name>
          <totalDrops>0</totalDrops>
          <dropsPerSec>0</dropsPerSec>
        </flow>
      </flowDrops>
      <cdbfiles>
        <file>
          <name>ARTCltSvr.cdb</name>
          <size>74551069696</size>
          <oldestDataTime>1475702400</oldestDataTime>
          <newestDataTime>1475862300</newestDataTime>
        </file>
        <file>
          <name>ARTSiteClt.cdb</name>
          <size>27521821696</size>
          <oldestDataTime>1475702400</oldestDataTime>
          <newestDataTime>1475862300</newestDataTime>
        </file>
        <file>
          <name>ARTSiteClt_lt.cdb</name>
          <size>41282231296</size>
          <oldestDataTime>1475704800</oldestDataTime>
          <newestDataTime>1475859600</newestDataTime>
        </file>
        <file>
          <name>ARTSiteSvr.cdb</name>
          <size>11009330176</size>
          <oldestDataTime>1475702400</oldestDataTime>
          <newestDataTime>1475862300</newestDataTime>
        </file>
        <file>
      <name>ARTSiteSvr_lt.cdb</name>
      <size>16513494016</size>
      <oldestDataTime>1475704800</oldestDataTime>
      <newestDataTime>1475859600</newestDataTime>
    </file>
    <file>
      <name>Hosts.cdb</name>
      <size>66190314496</size>
      <oldestDataTime>1475167500</oldestDataTime>
      <newestDataTime>1475862300</newestDataTime>
    </file>
    <file>
      <name>Hosts_lt.cdb</name>
      <size>99284970496</size>
      <oldestDataTime>1474480800</oldestDataTime>
      <newestDataTime>1475859600</newestDataTime>
    </file>
    <file>
      <name>RtpConv.cdb</name>
      <size>7142480896</size>
      <oldestDataTime>1475702280</oldestDataTime>
      <newestDataTime>1475862300</newestDataTime>
    </file>
    <file>
      <name>VoIPCalls.cdb</name>
      <size>14632324096</size>
      <oldestDataTime>1475702220</oldestDataTime>
      <newestDataTime>1475862300</newestDataTime>
    </file>
    <file>
      <name>CoreConv.cdb</name>
      <size>167216106496</size>
      <oldestDataTime>1474961760</oldestDataTime>
      <newestDataTime>1475862300</newestDataTime>
    </file>
    <file>
      <name>RtpMos.cdb</name>
      <size>244857856</size>
      <oldestDataTime>1475702280</oldestDataTime>
      <newestDataTime>1475862300</newestDataTime>
    </file>
    <file>
      <name>RtpMos_lt.cdb</name>
      <size>366785536</size>
      <oldestDataTime>1475704800</oldestDataTime>
      <newestDataTime>1475859600</newestDataTime>
    </file>
    <file>
      <name>DataSourceStats.cdb</name>
      <size>5052292096</size>
      <oldestDataTime>1474478100</oldestDataTime>
```

```
        <newestDataTime>1475862300</newestDataTime>
      </file>
      <file>
        <name>DataSourceStats_lt.cdb</name>
        <size>7577936896</size>
        <oldestDataTime>1474480800</oldestDataTime>
        <newestDataTime>1475859600</newestDataTime>
      </file>
      <file>
        <name>SiteStats.cdb</name>
        <size>13238864896</size>
        <oldestDataTime>1474478100</oldestDataTime>
        <newestDataTime>1475862300</newestDataTime>
      </file>
      <file>
        <name>SiteStats_lt.cdb</name>
        <size>19857796096</size>
        <oldestDataTime>1474480800</oldestDataTime>
        <newestDataTime>1475859600</newestDataTime>
      </file>
      <file>
        <name>SiteMatrix.cdb</name>
        <size>9058487296</size>
        <oldestDataTime>1474478100</oldestDataTime>
        <newestDataTime>1475862300</newestDataTime>
      </file>
      <file>
        <name>SiteMatrix_lt.cdb</name>
        <size>13587229696</size>
        <oldestDataTime>1474480800</oldestDataTime>
        <newestDataTime>1475859600</newestDataTime>
      </file>
      <file>
        <name>AlarmMessages.cdb</name>
        <size>15503236096</size>
        <oldestDataTime>0</oldestDataTime>
        <newestDataTime>0</newestDataTime>
      </file>
      <file>
        <name>MDIfStats_lt.cdb</name>
        <size>1464134656</size>
        <oldestDataTime>0</oldestDataTime>
        <newestDataTime>0</newestDataTime>
      </file>
    </cdbfiles>
  </systemInfo>
</nam-response>
```

NTP Time

The NTP Time API allows getting and setting the NTP configuration. It also allows retrieval of the current SECPA system time.

API Overview

URL	Method	Description
/nbi/nbi-ntp	GET	Get NTP configuration and current SECPA system time.
/nbi/nbi-ntp	POST	Set NTP configuration.

Get NTP Configuration and System Time

Method	GET
URI	/nbi/nbi-ntp
Description	Get NTP configuration and current SECPA system time.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), failed(-1)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <ntp-settings>
    <server>ntp.esl.cisco.com</server>
    <server>10.81.254.131</server>
    <region>America</region>
    <zone>Los_Angeles</zone>
  </ntp-settings>
  <time>2013-Oct-11, 13:02:40 PDT</time>
</nam-response>
```

Set NTP Configuration

Method	POST
URI	/nbi/nbi-ntp
Description	Set NTP configuration, consisting of NTP server address(es) and local time zone. At least one NTP server address must be specified; a maximum of two are allowed. The time zone setting is optional. The SECPA checks that an NTP server is present at each specified address; if not, the requested settings will not be saved, and the API call will return an error. After a successful request, the SECPA GUI and REST API will be unresponsive for 30-60 seconds while the SECPA software reinitializes.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	Successful(0), NTP Server Response Failed(-1), Server Internal Error(-2)

Sample Request

```
<ntp-settings>
  <server>ntp.esl.cisco.com</server>
  <server>10.81.254.131</server>
  <region>America</region>
  <zone>Los_Angeles</zone>
</ntp-settings>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

Monitoring Services

The Monitoring Services API allows various monitoring services provided by the SECPA to be enabled or disabled.

API Overview

URL	Method	Description
/nbi/nbi-monitor-service	GET	Get monitoring configuration.
/nbi/nbi-monitor-service	POST	Set monitoring configuration.

Get Monitoring Configuration

Method	GET
URI	/nbi/nbi-monitor-service
Description	Get monitoring configuration.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500), notImplemented(501)
SECPA Status Code(s)	successful(0), internalError(-1), badParameter(-3), invalidID(-14)

Sample Request

This operation does not require a request body.

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <monitor-services>
    <response-time-service>enabled</response-time-service>
    <response-time-filter>enabled</response-time-filter>
    <rtp-service>enabled</rtp-service>
    <url-service>disabled</url-service>
    <voice-signaling-service>enabled</voice-signaling-service>
  </monitor-services>
</nam-response>
```

Set Monitoring Configuration

Method	POST
URI	/nbi/nbi-monitor-service
Description	Set monitoring configuration.

HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	successful(0), failed(-1)

Sample Request

```
<monitor-services>
  <response-time-service>enabled</response-time-service>
  <response-time-filter>enabled</response-time-filter>
  <rtp-service>enabled</rtp-service>
  <url-service>disabled</url-service>
  <voice-signaling-service>enabled</voice-signaling-service>
</monitor-services>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
</nam-response>
```

CSV Data Query

The CSV Data Query API provides access to the Circular Database (CDB) tables that the SECPA uses to store various aggregated statistics and other records. Each CDB table is allocated a fixed amount of space, enough to store data for a fixed number of days. Once a table is filled to capacity, each subsequent entry overwrites the oldest entry in the table. Each table has an associated **aggregation interval**, which determines the granularity of the data stored in that table (this interval can be adjusted using the SECPA GUI or CLI). For example, if a table has an aggregation interval of 60 seconds, then every 60 seconds, the SECPA software adds a new table entry with metrics summarizing (aggregating) the data collected over that 60 second period.

CDB tables are modeled after traditional relational database tables. Each CDB table is defined by a set of columns, each with an associated name, data type, and unit of measure. Each data record is represented by a table row with a value under each column. Certain columns are designated as keys; no two table rows can have the values of all the key fields be identical. The column definitions of a CDB table comprise its **schema**. An XML representation of the CDB table schemas is included with the SECPA image; it can be downloaded at http://`secpa-host`/admin/cdb-schema.php, where `secpa-host` denotes the hostname or IP address of the SECPA.

Queries against the CDB are written using the subset of SQL described below. Many statistics presented by the SECPA GUI are actually computed using such SQL queries, so it is possible to compute similar statistics for external use by using the CSV Data Query API. The SQL queries take the following general form:

```
SELECT {* | col_expr_1, col_expr_2, ..., col_expr_N}
FROM cdb_table
WHERE TIME >= start_time AND TIME <= end_time [AND ...]
[GROUP BY col_name]
[ORDER BY col_expr [ASC | DESC]]
[LIMIT max_rows [, offset]]
```

Note the following:

- The SELECT clause is required. The select list can be either the `*` (asterisk) character to denote all columns in the table, or it can be a combination of column names and column expressions involving arithmetic operators and/or aggregate functions. The aggregate functions supported are COUNT, MAX, MIN, and SUM. The operators supported are `+` (addition), `-` (subtraction), `*` (multiplication), and `/` (division).
- The CDB query engine supports only SELECT statements. Other types of SQL statements like INSERT, UPDATE, and DELETE are not supported.
- The FROM clause is required, and supports exactly one table name. Cartesian product and join operations are not supported.
- The WHERE clause is required. The `start_time` and `end_time` specifiers are required, and must appear at the beginning of the WHERE clause. Each time value is a standard Unix time (i.e., number of seconds since the Unix epoch); most programming languages include standard library routines for working with Unix times. After the time specifiers, additional AND expressions referring to other columns may be added if desired.
- The GROUP BY clause is optional, and has the same behavior as in standard SQL. However, it is limited to one column name, which must also be the first column that appears in the SELECT list. Also, note that the HAVING clause is not supported.
- The ORDER BY clause is optional, and has the same behavior as in standard SQL. However, it is limited to one column name or expression, which must also be present somewhere in the SELECT list. The ASC or DESC keywords are optional as well; if neither is provided, the sort order defaults to ascending (i.e., smaller values first).
- The LIMIT clause is optional. If present, the `max_rows` parameter is required, and denotes the maximum number of rows to return. The `offset` parameter is optional. If provided, the first `offset` number of rows are omitted from the result set. For example, `LIMIT 10, 5` would omit the first 5 rows, thus yielding the 6th through 15th rows in the result set.

A full description of the syntax and semantics of SQL SELECT statements is beyond the scope of this guide. For additional background on writing queries, it may be helpful to consult an SQL book or tutorial.

Here are some concrete examples (note that appropriate values must be substituted for `start_time` and `end_time`):

- Top 10 applications:

  ```
  SELECT appId, SUM(octets)
  FROM DataSourceStats
  WHERE TIME >= start_time AND TIME <= end_time
  GROUP BY appId
  ORDER BY SUM(octets) DESC
  LIMIT 10, 1
  ```

  To look up the protocol associated with a given appId, use the Applications API.

- Top 10 hosts (by in + out traffic):

  ```
  SELECT host, SUM(inOctets), SUM(outOctets), SUM(inOctets)+SUM(outOctets)
  FROM Hosts
  WHERE TIME >= start_time AND TIME <= end_time
  GROUP BY host
  ORDER BY SUM(inOctets)+SUM(outOctets) DESC
  LIMIT 10, 1
  ```

The full list of CDB tables is shown below. Note that tables whose names carry the "_lt" suffix contain "long-term" data. Such tables have the same schema as their shorter-term counterparts, but have longer aggregation intervals. Thus, the metrics they maintain cover a longer time period, but with less detail.

- **ARTCltSvr** — Application Response Times (Client-Server)
- **ARTSiteClt** / **ARTSiteClt_lt** — Application Response Times (Site-Client)
- **ARTSiteSvr** / **ARTSiteSvr_lt** — Application Response Times (Site-Server)
- **AlarmMessages**
- **CoreConv** — Core conversations
- **DataSourceStats** / **DataSourceStats_lt**
- **Hosts** / **Hosts_lt**
- **MDIfStats** / **MDIfStats_lt** — Managed device interface statistics
- **RtpConv** — Real-time Transport Protocol conversations
- **RtpMos** / **RtpMos_lt** — Real-time Transport Protocol (RTP) Mean Opinion Score (MOS)
- **SiteMatrix** / **SiteMatrix_lt**
- **SiteStats** / **SiteStats_lt**
- **VoIPCalls** — Voice over IP (VoIP) calls

The SECPA CLI's `show cdb` command can be used to examine the properties of each CDB table.

When developing queries, it may be useful to experiment using the CDB query test page, which can be accessed at http://`secpa-host`/cdb. This page accepts an SQL query as input, returning the result set in a formatted tabular form as output.

Note that the CDB table schemas are primarily designed for use in implementing built-in SECPA functionality. The schemas are considered implementation details; as such, there is no guarantee that they will remain stable across SECPA software releases. Therefore, developers should use the CSV Data Query API only if they are willing to update code as necessary to match future schema updates.

API Overview

URL	Method	Description
/nbi/nbi-csvquery	POST	Query a CDB table for records matching specified parameters.

Query CDB Table

Method	POST
URI	/nbi/nbi-csvquery
Description	Query a CDB table for records matching specified parameters.
HTTP Normal Response Code(s)	successful(200)
HTTP Error Response Code(s)	unauthorized(401), badRequest(400), forbidden(403), notFound(404), internalError(500)
SECPA Status Code(s)	Successful(0), Invalid SELECT Column(-1), Invalid SELECT Syntax(-2), Invalid FROM Syntax(-3), Invalid FROM Table(-4), Invalid WHERE Column(-5), Invalid WHERE Syntax(-6), Invalid WHERE Value(-7), Invalid GROUPBY Column(-8) Invalid GROUPBY Syntax(-9), Invalid GROUPBY Prefix(-10), Invalid ORDERBY Column(-11), Invalid ORDERBY Syntax(-12), Invalid ORDERBY Spec(-13), Invalid LIMIT Syntax(-14), Invalid Time Range(-15), Invalid Syntax(-16), Table Schema Error(-17), Out Of Resource(-18)

Sample Requests

In the examples below, note that XML special characters must be written in their XML entity form. That is, `>` is written as `&gt;`, and `<` is written as `&lt;`. This escaping is needed for the XML to be well-formed. While not specific to CSV Data Query requests, other SECPA REST API requests do not typically involve the use of characters that require escaping.

Top 10 Applications:

```
<query-data>
  <query>
    SELECT appId, SUM(octets)
    FROM DataSourceStats
    WHERE TIME &gt;= 1384996848 AND TIME &lt;= 1384999848
    GROUP BY appId
    ORDER BY SUM(octets) DESC
    LIMIT 10, 1
  </query>
</query-data>
```

Top 10 Hosts:

```
<query-data>
  <query>
    SELECT host, SUM(inOctets), SUM(outOctets), SUM(inOctets)+SUM(outOctets)
    FROM Hosts
    WHERE TIME &gt;= 1384996848 AND TIME &lt;= 1384999848
    GROUP BY host
    ORDER BY SUM(inOctets)+SUM(outOctets) DESC
    LIMIT 10, 1
  </query>
</query-data>
```

Top 10 Slowest Client Application Response Times for HTTP (millisecond resolution):

```
<query-data>
  <query>
    SELECT client, clientSite, serverSite, maxRspTime, appId
    FROM ARTSiteClt_lt
    WHERE TIME &gt;= 1386202262 AND TIME &lt;= 1386206462 AND appId = 50331728 <!--
HTTP -->
    GROUP BY client
    ORDER BY maxRspTime DESC
    LIMIT 10, 1
  </query>
</query-data>
```

Top 10 Slowest Client Application Response Times for HTTP (microsecond resolution):

```
<query-data>
  <time-unit>microseconds</time-unit>
  <query>
    SELECT client, clientSite, serverSite, maxRspTime, appId
    FROM ARTSiteClt_lt
    WHERE TIME &gt;= 1386202262 AND TIME &lt;= 1386206462 AND appId = 50331728 <!--
HTTP -->
    GROUP BY client
    ORDER BY maxRspTime DESC
    LIMIT 10, 1
  </query>
</query-data>
```

Sample Responses

Top 10 Applications:

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <query-data>
    <totalEntries>13</totalEntries>
    <intervalInSecs>60</intervalInSecs>
    <column-name>appId,octets</column-name>
    <column-type>integer,uint64</column-type>
    <row>50331728,166460258938</row>
    <row>50331668,12271163398</row>
    <row>218103869,6157770732</row>
    <row>50331669,1526531745</row>
    <row>50336708,5381852</row>
    <row>301991942,2922504</row>
    <row>100663641,2748547</row>
    <row>201326647,2256400</row>
    <row>218103809,1061160</row>
    <row>302014465,56653</row>
  </query-data>
</nam-response>
```

Top 10 Hosts:

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
  <operation-result>
    <status>0</status>
    <description>Successful</description>
  </operation-result>
  <query-data>
    <totalEntries>1617</totalEntries>
    <intervalInSecs>60</intervalInSecs>
    <column-name>host,inOctets,outOctets,inOctets</column-name>
    <column-type>netaddr,uint64,uint64,uint64</column-type>
    <row>50.6.11.61,485962147,28475496,514437643</row>
    <row>50.6.11.65,485942812,28476234,514419046</row>
    <row>50.6.11.34,485891816,28472047,514363863</row>
    <row>50.6.11.48,485890407,28468995,514359402</row>
    <row>50.6.11.32,485872013,28468749,514340762</row>
    <row>50.6.11.33,485868124,28466679,514334803</row>
    <row>50.6.11.44,485819660,28467973,514287633</row>
    <row>50.6.11.57,485819504,28464399,514283903</row>
    <row>50.6.11.46,485818760,28463471,514282231</row>
    <row>50.6.11.55,485817533,28462868,514280401</row>
  </query-data>
</nam-response>
```

Top 10 Slowest Client Application Response Times for HTTP (millisecond resolution):

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
    <operation-result>
        <status>0</status>
        <description>Successful</description>
    </operation-result>
    <query-data>
        <totalEntries>375</totalEntries>
        <intervalInSecs>3600</intervalInSecs>
        <column-name>client,clientSite,serverSite,maxRspTime,appId</column-name>
        <column-type>netaddr,integer,integer,integer,integer</column-type>
        <row>50.6.11.136,1,1,2502,50331728</row>
        <row>50.6.10.220,1,1,1109,50331728</row>
        <row>50.6.10.123,1,1,1076,50331728</row>
        <row>50.6.10.31,1,1,1067,50331728</row>
        <row>50.6.11.105,1,1,1053,50331728</row>
        <row>50.6.10.237,1,1,1052,50331728</row>
        <row>50.6.11.85,1,1,1052,50331728</row>
        <row>50.6.10.142,1,1,1052,50331728</row>
        <row>50.6.10.11,1,1,1052,50331728</row>
        <row>50.6.10.90,1,1,1052,50331728</row>
    </query-data>
</nam-response>
```

Top 10 Slowest Client Application Response Times for HTTP (microsecond resolution):

```
<?xml version="1.0" encoding="UTF-8"?>
<nam-response>
    <operation-result>
        <status>0</status>
        <description>Successful</description>
    </operation-result>
    <query-data>
        <totalEntries>375</totalEntries>
        <intervalInSecs>3600</intervalInSecs>
        <column-name>client,clientSite,serverSite,maxRspTime,appId</column-name>
        <column-type>netaddr,integer,integer,integer,integer</column-type>
        <row>50.6.11.136,1,1,2502168,50331728</row>
        <row>50.6.10.220,1,1,1109910,50331728</row>
        <row>50.6.10.123,1,1,1076682,50331728</row>
        <row>50.6.10.31,1,1,1067972,50331728</row>
        <row>50.6.11.105,1,1,1053042,50331728</row>
        <row>50.6.10.237,1,1,1052841,50331728</row>
        <row>50.6.11.85,1,1,1052456,50331728</row>
        <row>50.6.10.142,1,1,1052390,50331728</row>
        <row>50.6.10.11,1,1,1052315,50331728</row>
        <row>50.6.10.90,1,1,1052191,50331728</row>
    </query-data>
</nam-response>
```

Appendix A: Acronyms

Acronym	Description
API	Application Programming Interface
ART	Application Response Time (formerly known as IAP)
CDB	Circular Database (SECPA-specific implementation)
CLI	Command-Line Interface
CSV	Comma-Separated Values
DPI	Deep Packet Inspection
DSCP	Differentiated Services Code Point
ERSPAN	Encapsulated Remote SPAN
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IAP	Intelligent Application Performance (now known as ART)
IETF	Internet Engineering Task Force
LAN	Local Area Network
MD5	Message Digest 5 (cryptographic hash algorithm, specified in RFC 1321)
MIB	Management Information Base
MOS	Mean Opinion Score
SECPA	Security Packet Analyzer
NBAR	Network-Based Application Recognition (protocol classification engine)
NBAR2	Next-Generation NBAR
NBI	Northbound Interface (another term for the REST API)
NDE	NetFlow Data Export
PHP	PHP: Hypertext Preprocessor
RFC	Request for Comments (a series of IETF technical notes and specifications)
REST	Representational State Transfer (an architectural style for web services)
RSPAN	Remote SPAN
RTP	Real-time Transport Protocol
SPAN	Switched Port Analyzer (port mirroring)
SNMP	Simple Network Management Protocol

Acronym	Description
TACACS+	Terminal Access Controller Access-Control System Plus
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VLAN	Virtual LAN
WAN	Wide Area Network
W3C	World Wide Web Consortium
XML	eXtensible Markup Language