Security Management System External Interface Guide

Version 4.3.0 Patch 1

5998-2911
October 2016

Legal and notice information

TippingPoint Security Management System (SMS) External Interface Guide
Publication Part Number: 5998-2911

Contents

About this guide

The *Security Management System External Interfaces* provides information required to interact with the SMS using a Web API. This guide also provides information about the management information database (MIB) files and object identifiers (OIDs) used by the SMS.

Target audience

This guide is intended for technicians and maintenance personnel who are responsible for installing, configuring, and maintaining TippingPoint security systems and associated devices. Users should be familiar with networking concepts and with the following standards and protocols:

- TCP/IP
- UDP
- ICMP
- Ethernet
- Simple Network Time Protocol (SNTP)
- Simple Mail Transport Protocol (SMTP)
- Simple Network Management Protocol (SNMP)

Related documentation

A complete set of product documentation for the Security Management System is available online. The product document set generally includes conceptual and deployment information, installation and user guides, CLI command references, safety and compliance information, and release notes.

For information about how to access the online product documentation, refer to the *Read Me First* document in your product shipment.

Conventions

This information uses the following conventions.

Typefaces

TippingPoint uses the following typographic conventions for structuring information:

Convention	Element
Bold font	• Key names • Text typed into a GUI element, such as into a box • GUI elements that are clicked or selected, such as menu and list items, buttons, and check boxes. Example: Click **OK** to accept.
Italics font	Text emphasis, important terms, variables, and publication titles
`Monospace font`	• File and directory names • System output • Code • Text typed at the command-line
`Monospace, italic font`	• Code variables • Command-line variables
`Monospace, bold font`	Emphasis of file and directory names, system output, code, and text typed at the command line

Messages

Messages are special text that is emphasized by font, format, and icons.

⚠**Warning!** Alerts you to potential danger of bodily harm or other potential harmful consequences.

△**Caution:** Provides information to help minimize risk, for example, when a failure to follow directions could result in damage to equipment or loss of data.

Note: Provides additional information to explain a concept or complete a task.

Important: Provides significant information or specific instructions.

Tip: Provides helpful hints and shortcuts, such as suggestions about how to perform a task more easily or more efficiently.

Contacting support

Contact the TippingPoint Technical Assistance Center (TAC) by using any of the following options.

Email support

tippingpoint.support@trendmicro.com

Phone support

North America: +1 866 681 8324

International: See *https://tmc.tippingpoint.com*

SMS Web Services

The SMS Web Services API enables you to access the SMS schema and data dictionary, and use `HTTP GET` and `POST` methods to retrieve data for managed devices, defined policies, Digital Vaccines, and policy notifications that are captured by the SMS.

The SMS Web Services interface provides a means to capture historical data for your TippingPoint network. By default, this interface is enabled for the SMS system.

Authentication and authorization

The SMS supports two authentication methods for the web API:

- Parameters encoded in the URL (`smsuser/smspass`).
- HTTP Basic Authentication (the `–u <username> –basic options on curl`).

Note: As part of the URL, passwords are sent in plain text, so these APIs should only be used on a trusted network.

By default, authentication is required for web access. You can change the authentication for web access through the command line interface or through the SMS client (see "System Preferences" in the *Security Management System User Guide*).

Note: Authentication is based on the SMS user accounts established for SMS client access. A SuperUser can use capabilities to grant SMS user accounts access. To configure access, use the Role wizard in the Admin workspace of the SMS client. For more information, see the *Security Management System User Guide*.

API definition

To submit a request operation to the SMS Web Services interface, send a URL that matches the following format:

`http[s]://<sms_server>/dbAccess/tptDBServlet?<parameters>`

Parameter	Variable	Sub-Parameter	Variable
method	Status		
	Version		
method	Schema	database	Oracle, MySQL (default) *Only valid for sql format*
method	DataDictionary	table (optional)	SEVERITY *Default is all variables* PRODUCT_CATEGORY TAXONOMY_MAJOR TAXONOMY_MINOR TAXONOMY_PROTOCOL TAXONOMY_PLATFORM ALERT_TYPE THRESHOLD_UNITS DEVICE SEGMENT SEGMENT_GROUP VIRTUAL_SEGMENT PROFILE ACTIONSET SIGNATURE POLICY

Parameter	Variable	Sub-Parameter	Variable
			QUARANTINE_NETWORK_DEVICES PROFILE_INSTALL_INVENTORY
method		format (optional)	sql (default), csv, xml
		mode (optional)	• insert (default) *only valid for sql format* • update • replace *only works with MySQL*
method	GetData	table	ALERTS DDOS_STATS QUARANTINE_HOSTS RATELIMIT_STATS THRESHOLD_STATS
method	GetData	begin_time	Integer *Time is expressed as the number of milliseconds since 01-01-1970 00:00:00 GMT*
method	GetData	end_time	Integer *Time is expressed as the number of milliseconds since 01-01-1970 00:00:00 GMT*
method	GetData	format (optional)	csv (default), sql, xml
method	GetData	limit (optional)	Integer *This is the maximum number of values returned. By default, all values are retuned.*

Parameter	Variable	Sub-Parameter	Variable
method	GetOldestRecord GetNewestRecord	table	ALERTS DDOS_STATS QUARANTINE_HOSTS RATELIMIT_STATS THRESHOLD_STATS

API usage

Use the Web Services interface according to the following sequence:

1. Use the Schema method to retrieve the schema definition. Apply the returned data to user-defined database.
2. Use the DataDictionary method to retrieve supporting data. Apply the returned data to database. You may repeat this step as needed, such as to create new profiles and activate new Digital Vaccines.
3. Continuously use the GetData method, and import the event data into the database.

Version

Use the `Version` method to obtain the version number of the SMS Web API you are using. The following example shows the URL format for the `GET` request.

```
http[s]://<sms_server>/dbAccess/tptDBServlet?method=Version
```

The SMS returns a version number if the request is successful.

Status

Use the `Status` method to determine whether Web Services support is enabled and running. The following example demonstrates the `GET` request to retrieve status.

```
http[s]://<sms_server>/dbAccess/tptDBServlet?method=Status
```

If Web Services support is up and running, the SMS returns the message *OK*.

If the Web server is enabled but Web Services support is *not* enabled, your response (in a Web browser window) is a Page Not Found message. Otherwise, a network timeout occurs due to no service.

Schema

Use the `Schema` method to obtain SMS database schema information. The following example demonstrates the `GET` request to retrieve the database schema information.

```
http[s]://<sms_server>/dbAccess/tptDBServlet?method=Schema
```

The SMS returns the schema information in Oracle 8i or MySQL 4.0 compliant data definition language (DDL) statements.

DataDictionary

Use the `DataDictionary` method to obtain SMS Data Dictionary information in a specific format. The following example shows the URL format to get `DataDictionary` information.

```
http[s]://<sms_server>/dbAccess/tptDBServlet?
method=DataDictionary&format=<sql>
```

The `format` parameter in the example is optional; if you do not specify format, the SMS returns `DataDictionary` information in SQL '92 compliant statements.

Valid format values include the following options:

- **sql** — SQL '92 compliant DML statements (default)
- **csv** — Comma-delimited file listing of the table data
- **xml** — XML for each table

For more information on the XML response to a `DataDictionary` request, see *DataDictionary XML response* on page 25.

GetData

Use the `GetData` method to request data from specific tables, specify certain parameters as constraints, and specify a format. The following examples show the URL format to request certain data using the `GetData` method.

GetData from ALERTS table, Begin/End times, CSV format

```
http[s]://<sms_server>/dbAccess/tptDBServlet?
method=GetData&table=ALERTS&begin_time=1&end_time=1162252800000&format=<csv>
```

GetData from THRESHOLD_STATS table, Begin/End times, SQL format

```
http[s]://<sms_server>/dbAccess/tptDBServlet?method=GetData&
table=THRESHOLD_STATS&begin_time=1&end_time=1162252800000&format=<sql>
```

GetData from RATE_LIMIT_STATS table, Begin/End times, XML format

```
http[s]://<sms_server>/dbAccess/tptDBServlet?method=GetData&
table=RATE_LIMIT_STATS&begin_time=1&end_time=1162252800000&format=<xml>
```

The `format` parameter in the examples are optional; if you do not specify format, the SMS returns your requested data in SQL '92 compliant statements.

Valid format values include the following options:

- **sql** — SQL '92 compliant DML statements (default)
- **csv** — Comma-delimited file listing of the table data
- **xml** — XML for each table

SMS Schema Reference

The SMS Schema Reference information describes the table and column relationships as defined by the SMS external database schema. The schema has referential integrity constraints, as well as unique and foreign keys.

Unique keys constrain the table to only one specific entry of that type; this ensures the uniqueness of each entry in that table. The exception is the ALERTS table. The ALERTS table uses a multi-column unique index. For more information, see *ALERTS table* on page 29.

Foreign keys enforce a binding between tables that have a relation. Foreign keys are labeled with a (FK) symbol and have a dotted line between that column and the table it is constrained to have an matching entry.

DataDictionary

The SMS `DataDictionary` is a repository of data related to profiles, devices, segments, and virtual segments. The `DataDictionary` includes the following tables:

ACTIONSET table

An ACTIONSET record is one defined by the user and applied to a POLICY. The ACTIONSET has a descriptive name that can help determine the action that is taken when a POLICY is triggered. For RATELIMIT ACTIONSETs, the RATE column has a value specifying the RATE to be applied. This table is not expected to grow by many entries. It is a relatively small table.

Column	Description
ID	Unique identifier for the record entry; use this column to join from other tables
NAME	Descriptive name for the ACTIONSET
RATE	RATELIMIT value applied to this ACTIONSET
FLOW_CONTROL	Traffic flow indicator (ALLOW, DENY, TRUST, and RATE)

ALERT_TYPE table

A simple table that gives descriptive names for ALERTS. The table should not grow, but may have new types added in future releases.

Column	Description
ID	Unique identifier for this record
NAME	Descriptive name for the entry

CATEGORY table

The CATEGORY table maintains the names used for SIGNATURE categories. The SIGNATURE table contains a number that is joined to the ID field in this CATEGORY table. The NAME field is the descriptive text for the CATEGORY.

Column	Description
ID	Unique identifier for the record entry; use this column to join from other tables
NAME	Descriptive name for the CATEGORY

DEVICE table

This table contains a record for each of the devices being managed. This table is not expected to grow by many entries. It is a relatively small table.

Column	Description
ID	Unique identifier for the table entry
SHORT_ID	Lookup identifier for the table entry
NAME	Descriptive name for the device provided during device installation
MODEL	String that represents the model of the device
SERIAL_NUMBER	Alpha-numeric TippingPoint serial number
IP_ADDRESS	IP address for the management port for the device
LOCATION	Descriptive location text entered during device installation
DV_VERSION	Current version of the Digital Vaccine installed on the device; if the device is a Core Controller, this field is null
OS_VERSION	Current version of the TOS installed on the device
DEVICE_GROUP	Name of the group to which the device belongs
MANAGED	Boolean to show if the device is currently managed by the SMS

NOTICE_ACTION table

A simple table that gives descriptive names for ALERTS. The table should not grow, but may have new types added in future releases.

Column	Description
ID	Unique identifier for the record entry; use this column to join from other tables
NAME	Descriptive name for the entry

POLICY table

The POLICY table holds objects that are setup to determine what actions to take and behavior to have for a SIGNATURE trigger. This table is expected to grow based on the number of changes made to the PROFILE table entries. It is a relatively small table.

Column	Description
ID	Unique identifier for the table entry
PROFILE_ID	Identifier of the PROFILE object that contained this POLICY
SIGNATURE_ID	Identifier of the SIGNATURE this object is defining in a POLICY
ACTIONSET_ID	Identifier for the ACTIONSET applied to this object
DISPLAYNAME	Descriptive name for the POLICY, which is usually the same as the SIGNATURE referenced by SIGNATURE_ID; however, THRESHOLDS allow you to name the POLICY
MULTIPART_GROUP_ID	Identifier for SMS policy group

POLICY_GROUP_LOOKUP table

This table holds the mapping information for policy group information in SMS and in device. This table is used to find the policy information from the event/statistic that comes from the device.

Example

Select * from DDOS_STATS DS, POLICY_GROUP_LOOKUP PGL, POLICY POL
where DS.DEV_GROUP_ID = PGL.DEV_GROUP_ID and PGL.SMS_GROUP_ID =
POL.MULTIPART_GROUP_ID;

Column	Description
DEV_GROUP_ID	Identifier for the POLICY group in device
SMS_GROUP_ID	Identifier of the PROFILE group in SMS

PRODUCT_CATEGORY table

The PRODUCT_CATEGORY table maintains the names used for SIGNATURE categories. The SIGNATURE table contains a number that is joined to the ID field in this PRODUCT_CATEGORY table. The NAME field is the descriptive text for the PRODUCT_CATEGORY.

Column	Description
ID	Unique identifier for the record entry; use this column to join from other tables
NAME	Descriptive name for the PRODUCT_CATEGORY

PROFILE table

The PROFILE table is a container for your POLICY entries. You are able to name the PROFILE, make changes to the POLICY objects, and then distribute to a segment group. Table size depends on the number of PROFILEs you create in the SMS. It is a relatively small table.

Column	Description
ID	Unique identifier for the table entry
VERSION	Current version of the PROFILE

Column	Description
NAME	Descriptive name of the PROFILE
DESCRIPTION	Description of the PROFILE

PROFILE_INSTALL_INVENTORY table

The PROFILE_INSTALL_INVENTORY table is a container for items associated with PROFILE entries. Table size depends on the number of PROFILEs you create in the SMS. It is a relatively small table.

Column	Description
VIRTUAL_SEGMENT_ID	Lookup identifier for the virtual segment where the profile was distributed
PROFILE_ID	Lookup identifier for the profile details
PROFILE_VERSION	Profile version
DISTRIBUTE_ID	Lookup identifier for the distribution details
COMPLETE_TIME	Time the profile distribution completed; this value is in milliseconds since Jan. 1, 1970 00:00:00 GMT

QUARANTINE_NETWORK_DEVICES table

The QUARANTINE_NETWORK_DEVICES table contains the defined quarantine switches.

Column	Description
NAME	Descriptive name for the network device switch type
IP_ADDRESS	IP address for the switch

SEGMENT table

A SEGMENT record represents a physical SEGMENT on a DEVICE. It is a relatively small table and is only expected to grow when new devices are added to your network.

Column	Description
ID	Unique identifier for this record entry
DEVICE_ID	DEVICE to which this SEGMENT belongs
NAME	Descriptive name
IP_ADDRESS	OBSOLETE IP Address that may be given to this SEGMENT This value was used in Discovery services, which have been removed from the product
SLOT_INDEX	Internal chassis slot number; this number is always 3 for physical segments and 0 for virtual segments
SEGMENT_INDEX	For physical segments, the physical segment number; for virtual segments, this number is 0

SEGMENT_GROUP table

A SEGMENT_GROUP record represents a group of physical SEGMENTS. It is a relatively small table and is only expected to grow when new devices are added to your network.

Column	Description
ID	Unique identifier for this entry
NAME	Descriptive name for the SEGMENT GROUP provided during group creation

SEVERITY table

The SEVERITY table is a static table used to provide descriptive text for SEVERITY fields.

Column	Description
ID	Unique identifier for this entry
NAME	Name given to the SEVERITY

SIGNATURE table

The SIGNATURE table details the currently active Digital Vaccine package on the SMS for use with devices. The table grows as new Digital Vaccines are released, downloaded, and activated.

Column	Description
ID	Unique identifier for this entry
NUMBER	Integer number used to reference this SIGNATURE; The number is assigned by TippingPoint.
SEVERITY_ID	Identifier for the SEVERITY of this SIGNATURE. Join to SEVERITY.ID to obtain a descriptive name of the SEVERITY.
NAME	Name given to the SIGNATURE by TippingPoint
CLASS	Descriptive classification for the SIGNATURE
PRODUCT_CATEGORY_ID	Category ID from PRODUCT_CATEGORY table, provided by TippingPoint
PROTOCOL	Well-known PROTOCOL of which this SIGNATURE is part

Column	Description
TAXONOMY_ID	TAXONOMY classification
CVE_ID	Comma-separated list of CVE IDs that can be used to link to the CVE database See: *http://www.cve.mitre.org/*
BUGTRAQ_ID	Comma-separated list of BugTraq IDs that can be used to link to the BugTraq database See: *http://www.securityfocus.com*
DESCRIPTION	Descriptive text detailing this SIGNATURE. This text is informative information provided by TippingPoint
MESSAGE	Message that can be filled in with ALERTS.MESSAGE_PARMS values to create a dynamic message for this SIGNATURE

TAXONOMY_MAJOR table

The TAXONOMY_MAJOR table details the TippingPoint signature taxonomy major classifications. See the *TippingPoint Event Taxonomy* document for Taxonomy specifics.

Column	Description
ID	Unique identifier for this entry
NAME	Short name for the TAXONOMY_MAJOR entry
DESCRIPTION	Descriptive text for the TAXONOMY_MAJOR entry

TAXONOMY_MINOR table

The TAXONOMY_MINOR table details the TippingPoint signature taxonomy minor classifications.

Column	Description
ID	Unique identifier for this entry
MAJOR_ID	Identifier of the major classification ID to which this minor classification relates
DESCRIPTION	Descriptive text for the TAXONOMY_MINOR entry

TAXONOMY_PLATFORM table

The TAXONOMY_PLATFORM table details the TippingPoint signature platforms.

Column	Description
ID	Unique identifier for this entry
DESCRIPTION	Descriptive text for the TAXONOMY_PLATFORM entry

TAXONOMY_PROTOCOL table

The TAXONOMY_PROTOCOL table details the TippingPoint signature protocols.

Column	Description
ID	Unique identifier for this entry
DESCRIPTION	Descriptive text for the TAXONOMY_PROTOCOL entry

THRESHOLD_GROUP_LOOKUP table

This table holds the mapping information for threshold policy group information in SMS and in device. This table is used to find the policy information from the event/statistic that comes from the device.

Example

select * from THRESHOLD_GROUP_LOOKUP TGL, POLICY POL, THRESHOLD_STATS TS where TGL.SMS_GROUP_ID = POL.MULTIPART_GROUP_ID and TGL.DEV_GROUP_ID = TS.DEV_GROUP_ID;

Column	Description
DEV_GROUP_ID	Identifier for the POLICY group in device
SMS_GROUP_ID	Identifier for the POLICY group on SMS

THRESHOLD_UNITS table

The THRESHOLD_UNITS table defines the UNITS in which THRESHOLDS can be specified. This table is not expected to grow and has very few records.

Column	Description
ID	Unique identifier for this entry
NAME	Descriptive name for this UNIT entry

TPT_DEVICE table

An IPS entry. This table contains a record for each of the IPS's being managed.

Column	Description
ID	Unique identifier for this entry
SHORT_ID	Unique identifier for the table entry in integer format
DISPLAY_NAME	A descriptive name for the device provided by the end user during device installation

Column	Description
DEVICE_MODEL	A string that represents the IPS model
IP_ADDRESS	The IP address for the management port of this IPS
LOCATION	A descriptive location text entered by the user during device installation

TPT_PORT table

A TPT_PORT record represents a physical PORT on a DEVICE. It is a relatively small table and is only expected to grow when new IPS devices are added to your network.

Column	Description
ID	Unique identifier for this entry
DISPLAY_NAME	A descriptive name for the port

TPT_SEGMENT table

A SEGMENT record represents a physical SEGMENT on a DEVICE. It is a relatively small table and is only expected to grow when new IPS devices are added to your network.

Column	Description
ID	Unique identifier for this record entry
DEVICE_ID	The DEVICE which this SEGMENT belongs to
DEVICE_SHORT_ID	The short ID of DEVICE which this SEGMENT belongs to
DISPLAY_NAME	A descriptive name entered by the end user

Column	Description
IP_ADDRESS	OBSOLETE IP Address that may be given to this SEGMENT. This value was used in Discovery services which have been removed from the product
SEGMENT_SLOT	The internal chassis slot number. This number is always 3 for physical segments and 0 for virtual segments
SEGMENT_INDEX	For physical segments, the physical segment number. For virtual segments, this number is 0

VIRTUAL_SEGMENT table

A VIRTUAL_SEGMENT record represents a virtual physical SEGMENT on a DEVICE. It is a relatively small table and is only expected to grow when new devices are added to your network.

Column	Description
ID	Unique identifier for this record entry
DEVICE_ID	DEVICE to which this SEGMENT belongs
SEGMENT_GROUP_ID	SEGMENT GROUP to which this SEGMENT belongs
NAME	Descriptive name

DataDictionary XML response

When the SMS receives a valid, authenticated request using `DataDictionary` method, it can return an XML response. Specific response content depends on data you specify in the request. The following table lists the database tables that can be called by an external system, and describes the type of data included in the response.

Table name	Response
PROFILE_INSTALL_INVENTORY	Lists virtual segments that are enabled by IPS administrators. Each entry contains the following data: VIRTUAL_SEGMENT_ID PROFILE_ID PROFILE_VERSION DISTRIBUTE_ID COMPLETE_TIME See *PROFILE_INSTALL_INVENTORY table* on page 17 for more information.
DEVICE	Lists the IPS devices. Each entry contains the following data: ID SHORT_ID NAME MODEL SERIAL_NUMBER IP_ADDRESS LOCATION DV_VERSION OS_VERSION DEVICE_GROUP MANAGED See *DEVICE table* on page 14 for more information.

Table name	Response
SEGMENT	Lists physical segments that are enabled by IPS administrators. Each entry contains the following data: ID DEVICE_ID NAME IP_ADDRESS SLOT_INDEX SEGMENT_INDEX See *SEGMENT table* on page 18 for more information.
VIRTUAL_SEGMENT	Lists virtual segments that are defined by IPS administrators. Each entry in the list contains the following data: ID DEVICE_ID SEGMENT_GROUP_ID NAME See *VIRTUAL_SEGMENT table* on page 24 for more information.

XML response sample

The following is a sample XML response to a PROFILE_INSTALL_INVENTORY table request. This sample response includes information for the ten profile entries in the PROFILE_INSTALL_INVENTORY table.

Note that, while the response is formatted in XML, for historical reasons it was not designed to conform to the common practice of schema-based XML.

```
<?xml version="1.0" ?>
 <resultset>
  <table name="PROFILE_INSTALL_INVENTORY">
   <column name="VIRTUAL_SEGMENT_ID"type="Integer"/>
   <column name="PROFILE_ID"type="String"/>
   <column name="PROFILE_VERSION"type="String"/>
   <column name="DISTRIBUTE_ID"type="String"/>
   <column name="COMPLETE_TIME"type="Long"/>
```

```
<data>
 <r>
   <c>0</c>
   <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
   <c>85.4109</c>
   <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
   <c>1345218057347</c>
 </r>
 <r>
   <c>1</c>
   <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
   <c>85.4109</c>
   <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
   <c>1345218023621</c>
 </r>
 <r>
   <c>10</c>
   <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
   <c>85.4109</c>
   <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
   <c>1345218023621</c>
 </r>
 <r>
   <c>11</c>
   <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
   <c>85.4109</c>
   <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
   <c>1345218023621</c>
 </r>
 <r>
   <c>10</c>
   <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
   <c>85.4109</c>
   <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
   <c>1345218023621</c>
 </r>
 <r>
   <c>11</c>
   <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
   <c>85.4109</c>
   <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
   <c>1345218023621</c>
 </r>
 <r>
   <c>12</c>
   <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
   <c>85.4109</c>
   <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
   <c>1345218023621</c>
```

```
      </r>
      <r>
        <c>2</c>
        <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
        <c>85.4109</c>
        <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
        <c>1345218023621</c>
      </r>
      <r>
        <c>20</c>
        <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
        <c>85.4109</c>
        <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
        <c>1345218023621</c>
      </r>
      <r>
        <c>21</c>
        <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
        <c>85.4109</c>
        <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
        <c>1345218023621</c>
      </r>
      <r>
        <c>30</c>
        <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
        <c>85.4109</c>
        <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
        <c>1345218023621</c>
      </r>
      <r>
        <c>31</c>
        <c>8d577840-e7f1-11e1-7c4f-6eddc2a345a7</c>
        <c>85.4109</c>
        <c>96c829e0-e87a-11e1-7c4f-6eddc2a345a7</c>
        <c>1345218023621</c>
      </r>
    </data>
  </table>
</resultset>Security Management System (SMS) External Interfaces
```

Events Data

The following dynamic Events Data tables are used with the `GetData` variable:

ALERTS table

The ALERTS table contains information pertaining to the event that caused a POLICY to trigger. When an ACTIONSET is applied to a POLICY and it has a **Management Console** notification selected, it is put in the ALERTS table.

The primary key, a unique key, is a four column index, DEVICE_ID, ALERT_TYPE_ID, SEQUENCE_NUM, and END_TIME.

The table is expected to have a continuous growth pattern and contain millions of records. The data is retrieved by using the `method=GetData&table=ALERTS` parameter.

The following table lists the table columns:

Column	Description
SEQUENCE_NUM	Part of the ALERTS table unique index; it is a reference to a particular logs row entry counter. The ALERT_TYPE column defines the log being referenced. **Note:** This sequence number is not reliable as far as counting on it behaving as an ever increasing sequential number. It can be reset on the device and repeated for new events.
DEVICE_ID	Identifier for the DEVICE entry that sent the notification; it is the second part of the ALERTS table unique index.

Column	Description
	A foreign key to the DEVICE table was left off for the purpose of performance and due to the possibility that a DEVICE entry may not have been yet stored in the DEVICE table for this external database.
ALERT_TYPE_ID	The TYPE column is the third and final primary key constraint on the ALERTS table. This field can be joined to the ALERT_TYPE table for a descriptive name for this column.
POLICY_ID	Identifier used to map this alert to a POLICY table entry.
SIGNATURE_ID	Identifier used to map this alert to a SIGNATURE table entry.
BEGIN_TIME	Time at which the event was first started or previously logged. This value is in milliseconds elapsed since Jan. 1, 1970 00:00:00 GMT. When using notification aggregation, this value and the END_TIME typically are off by the number of minutes specified in the aggregation setting. The difference between BEGIN_TIME and END_TIME may be larger if a lot of time passes between attack events. When aggregation is turned off, the BEGIN_TIME usually is the same as the END_TIME.
END_TIME	Time at which the notification was logged and sent to the Management Console. This value is in milliseconds elapsed since Jan. 1, 1970 00:00:00 GMT. Subtracting BEGIN_TIME from END_TIME can determine the length of an attack if aggregation is being used. The difference between BEGIN_TIME and END_TIME might be unexpectedly large if a lot of time passes between attack events. **Note:** This is the column used when comparing with BEGIN_TIME and END_TIME fields in the `GetData` method.

Column	Description
HIT_COUNT	Counter displaying the number of times the event triggered before the notification was sent to the Management Console.
SRC_IP_ADDR	Source IP of the packet causing the notification. Numeric value of an IPv4 address, or the low-order 64 bits for an IPv6 address if SRC_IP_ADDR_HIGH is not NULL.
SRC_IP_ADDR_HIGH	Source IP of the packet causing the notification. Numeric value of high-order 64 bits for an IPv6 address.
SRC_PORT	Source port of the packet causing the notification.
DST_IP_ADDR	Destination IP of the packet causing the notification. Numeric value of an IPv4 address, or the low-order 64 bits for an IPv6 address if DST_IP_ADDR_HIGH is not NULL.
DST_IP_ADDR_HIGH	Destination IP of the packet causing the notification. Numeric value of high-order 64 bits for an IPv6 address.
DST_PORT	Destination port of the packet causing the notification.
VIRTUAL_SEGMENT_INDEX	Identifier for which device segment this alert was seen on.
PHYSICAL_PORT_IN	Device port on which the event was detected.
VLAN_TAG	VLAN identifier contained in the event.
SEVERITY	SEVERITY of the event. Usually corresponds to the SIGNATURE.SEVERITY column, joined by the SIGNATURE_ID column. A foreign key constraint to the SEVERITY table has been applied here.
PACKET_TRACE	Indicates if a packet trace is available on the device.

Column	Description
DEVICE_TRACE_BUCKET	Part of the device packet trace identifier.
DEVICE_TRACE_BEGIN_SEQ	Part of the device packet trace identifier.
DEVICE_TRACE_END_SEQ	Part of the device packet trace identifier.

MESSAGE_PARMS	Variable list of message parameters. This value can be tokenized and combined with the SIGNATURE.MESSAGE data to display a dynamic ALERT message.
	Join SIGNATURE_ID with SIGNATURE.ID to retrieve the SIGNATURE.MESSAGE data. The MESSAGE_PARMS string is a delimited string, the delimiter is the "|" character. The SIGNATURE.MESSAGE string contains place holders for these strings, the place holders are %1, %2, %n…. The tokenized MESSAGE_PARMS replaces the %n values based on there location in the string.
	Example
	`MESSAGE_PARMS=Steve|TippingPoint| developer SIGNATURE.MESSAGE=Hello, my name is %1. I am a %3 at %2.`
	Combining these two together would give:
	`Hello, my name is Steve. I am a developer at TippingPoint.`

QUARANTINE_ACTION	Quarantine action taken, either Added or Removed; used only in quarantine logs.
FLOW_CONTROL	Action taken by the action set: Permit, Rate Limit, or Trust.
ACTION_SET_UUID	Action set UUID; used only in rate limit logs.
ACTION_SET_NAME	Rate limit action; used only in rate limit logs.

Column	Description
RATE_LIMIT_RATE	Rate for rate limit logs; a numerical value followed by a unit. The unit can be Kbps or Mbps.
CLIENT_IP_ADDR	Long value of the Client IP address (**Capture Additional Event Information** must be enabled).
CLIENT_IP_ADDR_HIGH	Long value of the Client IP address (**Capture Additional Event Information** must be enabled). For IPV6 only.
XFF_IP_ADDR	Long value of the X-Forwarded-For IP address (**Capture Additional Event Information** must be enabled).
XFF_IP_ADDR_HIGH	Long value of the X-Forwarded-For IP address (**Capture Additional Event Information** must be enabled). For IPV6 only.
TCIP_IP_ADDR	Long value of the True-Client-IP address (**Capture Additional Event Information** must be enabled).
TCIP_IP_ADDR_HIGH	Long value of the True-Client-IP address (**Capture Additional Event Information** must be enabled). For IPV6 only.
URI_METHOD	Method of the URI.
URI_HOST	Host of the URI.
URI_STRING	URI string.
SRC_USER_NAME	User name on the source machine. User ID IP Correlation must be configured on the SMS to retrieve this information. User ID IP Correlation is a feature that enables the SMS to collect user authentication data directly and continuously from an Identity Agent device.

Column	Description
SRC_DOMAIN	Name of the source domain. User ID IP Correlation must be configured on the SMS to retrieve this information. User ID IP Correlation is a feature that enables the SMS to collect user authentication data directly and continuously from an Identity Agent device.
SRC_MACHINE	Name of the source machine. User ID IP Correlation must be configured on the SMS to retrieve this information. User ID IP Correlation is a feature that enables the SMS to collect user authentication data directly and continuously from an Identity Agent device.
DST_USER_NAME	User name on the destination machine. User ID IP Correlation must be configured on the SMS to retrieve this information. User ID IP Correlation is a feature that enables the SMS to collect user authentication data directly and continuously from an Identity Agent device.
DST_DOMAIN	Name of the destination domain. User ID IP Correlation must be configured on the SMS to retrieve this information. User ID IP Correlation is a feature that enables the SMS to collect user authentication data directly and continuously from an Identity Agent device.
DST_MACHINE	Name of the destination machine. User ID IP Correlation must be configured on the SMS to retrieve this information. User ID IP Correlation is a feature that enables the SMS to collect user authentication data directly and continuously from an Identity Agent device.

DDOS_STATS table

When using advanced DDOS policies, this data is accumulated from the DEVICE.

If you are using advanced DDOS, this table is expected to have a continuous growth pattern and contain millions of records.

The data is retrieved by using the `method=GetData&table=DDOS_STATS` parameter.

Column	Description
POLICY_ID	Identifier of the POLICY that was created to produce this DDOS data
STAT_TIME	Time the data was collected; this time is stored in milliseconds since Jan. 1, 1970 00:00:00 GMT
REJECT_SYNS	Number of rejected SYN requests for the stat period
PROXIED_CXNS	Number of proxied connections for the stat period
CPS_CXNS	Number of Connections Per Second over stat period
BLOCKED_CPS_CXNS	Number of blocked CPS in stat period
CFLOOD_CXNS	Number of Connection Flood connections in stat period
BLOCKED_CFLOOD_CXNS	Number of blocked Connection Flood connections in stat period

FIREWALL_BLOCK_LOG table

The FIREWALL_BLOCK_LOG table contains information pertaining to logs where traffic has been permitted by firewall rules that have logging enabled, including packets that were permitted by the content filtering configuration.

Column	Description
SEQUENCE_NUM	This field is a reference to a particular logs row entry counter
TPT_DEVICE_SHORT_ID	This is the identifier for the DEVICE entry that sent the notification

Column	Description
TIME	The time in which the event was first started. When using notification aggregation, this value and the TIME_END typically are off by the number of minutes specified in the aggregation setting. When aggregation is turned off, the BEGIN_TIME usually is the same as the TIME_END. This value is in milliseconds since Jan. 1, 1970 00:00:00 GMT.
TIME_END	The time in which the notification was sent to the Management Console. Subtracting BEGIN_TIME from TIME_END can determine the length of an attack if aggregation is being used. This value is in milliseconds since Jan. 1, 1970 00:00:00 GMT
HIT_COUNT	The number of times the firewall rule was applied
SRC_IP_ADDR	Source IP of the packet causing the notification
SRC_PORT	Source port of the packet causing the notification
DST_IP_ADDR	Destination IP of the packet causing the notification
DST_PORT	Destination port of the packet causing the notification
RULE_ID	Unique identifier for rule to monitor traffic between security zones
PROTOCOL_NAME	The packet type
PROTOCOL_NUMBER	The number associated with the protocol in the filter
PROTOCOL_TYPE	The protocol that was used to respond to the event
IN_ZONE_UUID	The security zone from which the attack originated
OUT_ZONE_UUID	The security zone from which the attack was targeted

Column	Description
PHYSICAL_PORT_IN	The device port on which the attack was detected
VLAN	The local VLAN that was targeted
CATEGORY	The type of traffic filter that was activated
SESSION_DURATION	The duration of the attack
URL	The URL that was associated with the attack, if applicable
URLINFO	Additional information relevant to the URL
SEVERITY	The severity of the attack

FIREWALL_TRAFFIC_LOG table

The FIREWALL_TRAFFIC_LOG table contains information pertaining to logs where traffic has been permitted by firewall rules that have logging enabled, including packets that were permitted by the content filtering configuration.

Column	Description
SEQUENCE_NUM	This field is a reference to a particular logs row entry counter
TPT_DEVICE_SHORT_ID	This is the identifier for the DEVICE entry that sent the notification
TIME_END	The time in which the notification was sent to the Management Console. Subtracting BEGIN_TIME from TIME_END can determine the length of an attack if aggregation is being used. This value is in milliseconds since Jan. 1, 1970 00:00:00 GMT
SRC_IP_ADDR	Source IP of the packet causing the notification

Column	Description
SRC_PORT	Source port of the packet causing the notification
DST_IP_ADDR	Destination IP of the packet causing the notification
DST_PORT	Destination port of the packet causing the notification
RULE_ID	Unique identifier for rule to monitor traffic between security zones
PROTOCOL_NAME	The packet type
PROTOCOL_NUMBER	The number associated with the protocol in the filter
IN_ZONE_UUID	The security zone from which the attack originated
OUT_ZONE_UUID	The security zone from which the attack was targeted
CATEGORY	The type of traffic filter that was activated
SESSION_DURATION	The duration of the attack
URL	The URL that was associated with the attack, if applicable
XFER_BYTES	The number of bytes transferred for this event
MESSAGE	A dynamic ALERT message

PORT_TRAFFIC_STATS table

This table contains information of traffic going through each port of IPS.

Column	Description
DEVICE_ID	Identifier for the DEVICE entry that sent the notification
PORT_ID	Identifier for the PORT entry that the traffic going through
SMS_TIME	SMS time in which the statistics get captured
DEVICE_TIME	Device SMS time in which the statistics get captured
IN_OCTETS	Device SMS time in which the statistics get captured
OUT_OCTETS	Total traffic going out the port

QUARANTINE_HOSTS table

The QUARANTINE_HOSTS table is where quarantine actions for devices and SMS actions are tracked.

The data is retrieved by using the `method=GetData&table=QUARANTINE_HOSTS` parameter.

Column	Description
ID	Unique identifier for the table entry
QUARANTINED_IP	IP address of the quarantined host
QUARANTINED_MAC	MAC address of the quarantined host
POLICY_NAME	Descriptive name for the policy that triggered the host quarantine
STATE	Current state of the host - UNQUARANTINED, QUARANTINED, INITIAL, or ERROR
AUTHORITY	Source of the quarantine state for the host

Column	Description
CREATE_TIME	Time the initial quarantine state was set
LAST_UPDATE	Time of the last quarantine state change

RATELIMIT_STATS table

When using RATELIMIT ACTIONSETs, this data is accumulated from the DEVICE.

If you are using RATELIMIT ACTIONSETs, this table is expected to have a continuous growth pattern and contain millions of records.

The data is retrieved by using the `method=GetData&table=RATELIMIT_STATS` parameter.

Column	Description
ACTIONSET_ID	Identifier of the ACTIONSET table entry for this record
STAT_TIME	Time this stat was recorded; the time is milliseconds since Jan. 1, 1970 00:00:00 GMT
DEVICE_ID	Identifier for the DEVICE
RATE	RATE in kbps
VALUE	Number of Bytes

THRESHOLD_STATS table

When using THRESHOLD policies, this data is accumulated from the DEVICE.

If you are using THRESHOLDs, this table is expected to have a continuous growth pattern and contain millions of records. The data is retrieved by using the `method=GetData&table=THRESHOLD_STATS` parameter.

Column	Description
POLICY_ID	Identifier of the POLICY that was created to produce this THRESHOLD data
STAT_TIME	Time the data was collected; time is stored in milliseconds since Jan. 1, 1970 00:00:00 GMT
VALUE	Stat value for the MEAS_UNIT at collection time
MEAS_UNIT	Identifier can be used to join with the THRESHOLD_UNITS table and obtain a descriptive name for the unit

Active Response

Active Response is a policy-based service in the SMS that reacts to its inputs in order to perform a set of actions. Its reactions, and the set of actions it takes, are based on the Active Response policies that have been configured.

A policy contains a set of actions that the Active Response service takes when the policy is triggered. A policy can be triggered in several ways: thresholding, manual triggering, Web service, or escalation of an IPS Quarantine action. Policies can be configured to include and/or exclude a set of IP addresses.

An external third party can initiate an Active Response via a simple Web API. This enables partners to interact with an TippingPoint secured environment and help to protect the network according to their own triggers or through some form of manual intervention.

Note: By default, no policies can be externally triggered. To enable external triggering, an Active Response policy must be configured to allow an SNMP trap or Web service to invoke the policy. In the SMS client, you can create or edit an Active Response policy in the Policies area of the Responder workspace. See the *SMS User Guide*.

To initiate an Active Response, the external system must use a URL that matches one of the following patterns:

- Create a Response (Quarantine URL)

```
http[s]://<sms_server>/quarantine/quarantine?ip=<target_ip>
&policy=<policy_name>&timeout=<minutes_to_quarantine>
&smsuser=<user_name>&smspass=<password parameters in the URL>
```

- Close a Response (Unquarantine URL)

```
http[s]://<sms_server>/quarantine/unquarantine?ip=<target_ip>
&smsuser=<user_name>&smspass=<password parameters in the URL>
```

Note: To close a response, either `ip` or `id` must be specified.

Parameters

Use the following parameters to initiate an Active Response operation.

Parameter	Description
`ip`	IP address for the target host. Required to create or close a response.

Parameter	Description
`id`	Response History ID that is displayed in the Response History table. To close a response, either IP or ID must be specified.
`policy`	Specific Active Response Policy to implement. The policy name is case sensitive and must match an existing SMS Active Response policy name. The **Allow an SNMP Trap or Web Service call to invoke this Policy** initiation setting must be enabled for this policy. This argument is not necessary to close a response and, if provided, is ignored.
`timeout`	Optional argument to specify the duration of response. The specified value overrides the default already in the policy. If no parameter is specified, the timeout value from the policy is used. This argument is not necessary to close a response and, if provided, is ignored.
`smsuser` `smspass`	Authenticated SMS user name and password. Required to initiate an Active Response operation. This applies to the Web interface as well, even if the Web authentication preference in the SMS is not turned on. **Note:** A password is not required, but you must specify the `smspass` parameter.

Using this interface

We recommend that you take the following steps to customize how you use this interface.

1. Use a unique user name and password for external interactions. Create a unique user name that is used only for this feature.

   Examples:

   `extqtime, webapi, webqtime`

2. Create customized policies specifically for this interface. An admin-level user can initiate via this interface. This helps to organize which policies are involved with any calls that happen externally.

Remote Profile Management

A *profile* is a collection of filters or rules that provide a method for setting up security configuration options for TippingPoint solutions. The Remote Profile Management API enables you to import, export, and manage an SMS profile through an HTTP interface instead of the SMS client. In addition, you can retrieve data for SMS managed devices, defined policies, Digital Vaccines, and policy notifications captured by the SMS.

The SMS Web API allows you to make changes on the SMS remotely; therefore, you are required to provide a valid user name and password as part of the API call.

Parameters can be passed to the SMS either by entering a URL in a Web browser or by using a command line tool for HTTP scripting. Command line tools make it possible to perform multiple operations using a script. One such open-source tool is cURL, which is available at *http://curl.haxx.se/*.

Certain Remote Profile Management servlets receive POST requests in XML format. In these instances, the XML schema and sample requests and responses are documented.

Authentication

A valid SMS user name and password are required to use the Web API. The user ID must exist on the SMS and have the permissions and the Profiles capabilities necessary for the tasks the user is to perform.

To export or import a profile, the user must be granted permission to that profile. For distributing a profile, the user account must have permission to the profile and to the segment group or segments that are the target of the distribution.

The user name and password may be specified as part of the URL or as part of the cURL invocation.

URL authentication

To specify authentication credentials as part of a URL, include the `smsuser` and `smspass` variables in the following pattern: `smsuser=<user_name>&smspass=<password_parameters>`

For example:

```
http[s]://<sms_server>/ipsProfileMgmt/exportProfile?
profileName=MyTestProfile&smsuser=TestUser&smspass=UserPassw0rd
```

Note: As part of the URL, passwords are sent in plain text, so these APIs should only be used on a trusted network.

HTTP Basic Authentication

The SMS supports HTTP Basic Authentication, and many HTTP client libraries support this. For example, you can do this with cURL using the `-u` option to specify the user name and password.

For example:

```
curl -u mySmsSuperUser:mypassword "https://<sms_server>/ipsProfileMgmt/
exportProfile?profileName=MyTestProfile"
```

Note: Only use HTTPS for API invocations to ensure that all credentials and data is encrypted in transit. The SMS SSL X509 certificate presented by the web browser by default can be customized or replaced in the SMS in the Admin workspace.

Traffic management filter

The Remote Profile Management API enables you to create traffic management filters. The following example shows the URI format to create a traffic management filter.

```
http[s]://<sms_server>/ipsProfileMgmt/createTrafficMgmt?
name=<filter_name>&profile=
<profile_name>&srcAddr=<ip_address>&destAddr=<ip_address>
```

The following table describes the parameters that are required to create a traffic management filter.

Parameter	Description
`name`	Name of the traffic management filter, names must be unique for each profile
`profile`	Name of the profile that contains the traffic management filter, the profile must already exist
`srcAddr`	Source address for the filter, valid value can be `any` or an IP address
`destAddr`	Destination address for the filter, valid value can be `any` or an IP address

The following table describes parameters that are optional. If a parameter is not specified, the default value is used.

Parameter	Description	Default
`direction`	Direction of filter, valid values are `AtoB`, `BtoA`, or `both`	`AtoB`

Parameter	Description	Default
`action`	Action set to use, valid values are restricted to allow, block, and trust. For rate limiting, use the rate-limit parameter	`block`
`rate-limit`	Rate limiting action set to use. The action set must already be defined and be set to rate limit	
`protocol`	Protocol to filter, valid values are `ip`, `ipv6`, `tcp`, `tcpv6`, `udp`, `udpv6`, `icmp`, and `icmpv6`	`ip`
`ipFragments`	Apply only to IP fragments, valid only when protocol is IP. Valid values are `true` and `false`	`false`
`icmptype`	ICMP type, valid only when protocol is `ICMP`. Valid values are 0-255	0
`icmpcode`	ICMP code, valid only when protocol is `ICMP`. Valid values are 0-255	0
`srcPort`	Source port to filter on, valid only when protocol is `TCP` or `UDP`. Valid values are `any` or 0-65535	0, which is *all ports*
`destPort`	Destination port to filter on, valid only when protocol is `TCP` or `UDP`. Valid values are `any` or 0-65535	0, which is *all ports*
`position`	Precedence of filter, valid values are 0-200	0, which uses the lowest unused value
`comment`	Comment for filter	
`state`	State of filter, valid values are `enable` and `disable`	`enabled`

Note: Parameter names and enumerated values are not case sensitive. If a parameter is specified multiple times, the behavior is unspecified.

Export a profile

The Remote Profile Management API enables you to export a profile to the SMS Web "Exports" directory or to a fully specified SMB or NFS server location.

The following example shows the URI pattern for an API call to export a profile. Note that location parameters are required only to export to SMB or NFS remote directories. Authentication parameters are required only if this information is not provided separately.

```
http[s]://<sms_server>/ipsProfileMgmt/exportProfile?[Export
Parameters]
```

```
[SMB Parameters|NFS Parameters][Authentication Parameters]
```

The following tables describe profile export parameters, SMB location parameters, NFS location parameters, and authentication parameters.

Modify a profile (export format)

When exporting profiles between two SMS clients, the files usually remain unmodified.

More advanced users can modify the files within a profile. For example, you can modify the XFF and URI settings in the `profile.xml` file. After you modify this file, remember to update the md5sum in the `sms-security-manifest` file before importing the profile file back in to the SMS.

Profile export parameters

Parameter	Description
`exportMethod` (optional)	Export destination: SMS HTTP server [default], smb, nfs
`profileName`	Name of profile to export
`profileVersion` (optional)	Version of profile to export; if `profileVersion` is not specified, the latest version of the profile is used

SMB location parameters

Parameter	Description
`remoteDirectory`	Remote SMB directory

Parameter	Description
`remoteFilename` (optional)	Remote filename (default: "profile_name.pkg")
`remoteServer`	SMB server
`userid`	SMB user ID
`password`	SMB password
`domain`	SMB domain

NFS location parameters

Parameter	Description
`remoteDirectory`	Remote NFS directory
`remoteFilename` (optional)	Remote filename (default: "profile_name.pkg")
`remoteServer`	NFS server

Authentication parameters

Parameter	Description
`smsuser`	SMS user name
`smspass`	Password for the SMS user

Examples

- Export MyProfile profile to the Export directory on the SMS server:

  ```
  http[s]://<sms_server>/ipsProfileMgmt/exportProfile?profileName=MyProfile
  ```

- Export Default profile to an SMB server:

  ```
  http[s]://<sms_server>/ipsProfileMgmt/exportProfile?
  exportMethod=SMB&remoteDirectory=savedProfiles&remoteServer
  =guest&password=guestpass&domain=MyExportDirectory&
  userid=CompanyXDomain&profileName=Default
  ```

- Export Default profile to an NFS server:

  ```
  http[s]://<sms_server>/ipsProfileMgmt/exportProfile?
  exportMethod=NFS&remoteDirectory=savedProfiles
  &remoteServer=MyExportDirectory&profileName=Default
  ```

Import a profile

The Remote Profile Management API enables you to import a profile from a local file to the SMS server. You do not need to specify a profile name because it is already specified within a profile package. The SMS supports the following import options for inspection reports:

- Replace an existing profile
- Add new settings to existing settings (no change to existing settings)
- Change existing settings

The version details section for the profile has an entry that indicates which profile the SMS profile was imported from, and includes a date that indicates when the update occurred. Previously, the description was updated.

Note: If you modify the profile file, you must maintain the same format. If you are importing a modified profile file, you must update the md5sum. For more information, see *Export a profile* on page 47.

Shared settings

Profiles include common configuration objects called shared settings, such as actions sets, notification contacts, and services.

If the imported profile includes policies or category settings that use a particular action set, the action set is added to the SMS. The SMS does not overwrite an existing action set with the same name, but rather renames the new action set by appending a number to the end of the file name, for example, “My Quarantine_2”.

A notification contact that is used by an action set is also imported and renamed, if necessary.

Existing port definitions for services on the SMS remain the same. If an imported profile includes a service with a port definition that differs from the existing service on the SMS, the service is added to the SMS

service list. You should review services any time a profile is imported from a different user or from a different environment.

URL method

The following example shows the URI format for importing a profile to the SMS. Authentication parameters are required only if this information is not provided separately.

Note: cURL is the preferred method to transfer profiles to the server.

```
http[s]://<sms_server>/ipsProfileMgmt/importProfile?[Profile Import Parameters]
```

```
[Authentication Parameters]
```

The following tables describe the profile import parameters and authentication parameters used for a profile import operation.

Profile import parameters

Parameter	Description
`filename`	File name of profile package file to import
`importAction`	Action to take: • `importAction` = “replace” (replace existing profile) • `importAction` = "add" (add any new settings to a new profile) • `importAction` = “combine_change” (replace existing settings with any new settings in the profile) • `importAction` = “combine_add” (combine existing profile with a new profile and add any new settings)
`targetProfileName`	If the `importAction` = “replace”, "combine_change", or "combine_add", then this is the name of the profile to be updated.

Parameter	Description
	Note: If the specified profile does not exist, a new one is created. If the specified profile *does* exist, its content is replaced with the content of the imported profile. In the SMS client, the profile information indicates that a profile was imported into the existing profile.
`replacedProfileName`	Name under which to save the replaced profile

Authentication parameters

Parameter	Description
`smsuser`	SMS user name
`smspass`	Password for the SMS user

Examples

Replace an existing profile:

```
curl -k -v -F "file=@</filepath/to/import.pkg>"
-F "importAction=replace"
-F "targetProfileName=<profile_name_on_sms_to_replace>"
-F "replacedProfileName=<profile_name_on_sms_to_replace>"
"https://<sms_server>/ipsProfileMgmt/importProfile?
smsuser=<sms_user>&smspass= <password>"
```

Add a new profile:

```
curl -k -v -F "file=@</filepath/to/import.pkg>"
-F "importAction=add"
-F "targetProfileName=<profile_name_on_sms_to_add>"
-F "replacedProfileName=<profile_name_on_sms_to_add>"
```

```
"https://<sms_server>/ipsProfileMgmt/importProfile?
smsuser=<sms_user>&smspass= <password>"
```

Combine with an existing profile and replace existing settings:

```
curl -k -v -F "file=@</filepath/to/import.pkg>"

-F "importAction=combine_change"

-F "targetProfileName=<profile_name_to_replace_existing_settings>"

-F "replacedProfileName=<profile_name_to_replace_existing_settings>"

"https://<sms_server>/ipsProfileMgmt/importProfile?
smsuser=<sms_user>&smspass= <password>"
```

Combine with an existing profile and add new settings:

```
curl -k -v -F "file=@</filepath/to/import.pkg>"

-F "importAction=combine_add"

-F "targetProfileName=<profile_name_to_combine_and_add_new_settings>"

-F
"replacedProfileName=<profile_name_to_combine_and_add_new_settings>"

"https://<sms_server>/ipsProfileMgmt/importProfile?
smsuser=<sms_user>&smspass= <password>"
```

Distribute profiles

The `distributeProfile` servlet allows you to initiate profile distribution to a single segment target or to a segment group.

The following example shows the URL format to initiate a profile distribution. Authentication parameters are required only if this information is not provided separately.

```
http[s]://<sms_server>/ipsProfileMgmt/distributeProfile?[Profile
Distribution

Parameters][Segment Group Target Parameters]|[Single Segment Target
Parameters]

[Authentication Parameters]
```

The following tables describe used to initiate a profile distribution.

Profile distribution parameters

Parameter	Description
`profileName`	Name of profile on SMS to distribute
`profileVersion` (optional)	Version of profile to distribute (latest version is used if not specified)
`distribPriority` (optional)	Priority of distribution on IPS: `high` [default] or `low`. If priority is not specified, `high` priority is used as a default

Segment group target parameter

Parameter	Description
`segmentGroupName`	Name of segment group that is target of distribution

Single segment target parameters

Parameter	Description
`deviceIpAddr`	IP Address of device, only required for single segment distributions
`segmentName`	Name of segment receiving distributed profile, only required for single segment distributions

Authentication parameters

Parameter	Description
`smsuser`	SMS user name
`smspass`	Password for the SMS user

Distribute a profile to a segment group

The following example shows the URL format to distribute a profile to a segment group.

```
http[s]://<sms_server>/ipsProfileMgmt/distributeProfile?
profileName=<profile_name>&segmentGroupName=
<SegmentGroupName>&smsuser=<sms_user>&smspass=<password>
```

Distribute a profile to a single segment

The following example shows the URL format to distribute a profile to a single segment.

```
http[s]://<sms_server>/ipsProfileMgmt/distributeProfile?
profileName=<profile_name>&deviceIpAddr=<device_ip_address>
&segmentName=<segment_name>
&smsuser=<sms_user>&smspass=<password>
```

Profile distribution XML schema

The Remote Profile Management API uses the following XML schema for profile distribution requests.

```
<?xml version="1.0" encoding="utf-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:simpleType name="uuid">
    <xs:restriction base="xs:string">
    <xs:pattern value="[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]
      {4}-[0-9a-f]{12}"/>
     </xs:restriction>
  </xs:simpleType>

 <xs:complexType name="idname">
    <xs:choice>
     <xs:element name="id" type="uuid"/>
     <xs:element name="name" type="xs:string"/>
    </xs:choice>
 </xs:complexType>

 <xs:element name="distribution">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="profile" minOccurs="1"maxOccurs="1">
    <xs:complexType>
      <xs:attribute name="id" type="uuid"/>
```

```
      <xs:attribute name="name" type="xs:string"/>
      <xs:attribute name="version" type="xs:string" use="required"/>
    </xs:complexType>
 </xs:element>

 <xs:element name="priority" minOccurs="0">
   <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="high"/>
      <xs:enumeration value="low"/>
    </xs:restriction>
   </xs:simpleType>
 </xs:element>

<xs:element name="segmentGroup" type="idname" minOccurs="0"
   maxOccurs="unbounded"/>
 <xs:element name="virtualSegment" minOccurs="0"
    maxOccurs="unbounded">
   <xs:complexType>
     <xs:sequence>
      <xs:element name="id" type="uuid"/>
     </xs:sequence>
    </xs:complexType>
 </xs:element>

 <xs:element name="device" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
   <xs:sequence>
    <xs:choice>
     <xs:element name="id" type="uuid"/>
     <xs:element name="shortID" type="xs:positiveInteger"/>
     <xs:element name="name" type="xs:string"/>
     <xs:element name="ipAddress" type="xs:string"/>
    </xs:choice>
     <xs:element name="virtualSegment" type="idname"
       maxOccurs="unbounded"/>
   </xs:sequence>
   </xs:complexType>

</xs:element>
 </xs:sequence>
  </xs:complexType>
  </xs:element>
</xs:schema>
```

XML elements

The following table describes XML elements in the profile distribution request schema.

Element	Value	Definition
`profile`	NULL element with these attributes: • id • name • version	IPS profile identified by an id expressed in UUID format, a name string and a version number string
`priority`	String, either `high` or `low`	High or low value indicating the priority for distributing the profile to the managed IPS devices
`segmentGroup`	String	ID string expressed in UUID format or a name string to specify the group of segments for the profile distribution
`virtualSegment`	String	ID string expressed in UUID format to specify a virtual segment
`device/id` (device)	String	Internal ID assigned to the device expressed in UUID format
`device/shortID`	Positive integer	Internal number assigned to the device
`device/name`	String	Name of the device
`device/ipAddress`	String	IP address string of the device
`device/virtualSegment`	String	ID string expressed in UUID format or a name string to specify a virtual segment on the device

Retrieve profile distribution status

The `distributionStatus` servlet returns the status of profile distributions specified in a `POST` request. You can use status requests to determine the success or failure and the duration of a distribution session. Note, however, that actual percent-complete progress and predicted end-time are not available.

A profile distribution status request must include at least one Distribution ID. A request can also include Device Name, Device ID, Device ShortID and Device IP Address.

The following example shows the URL format to request profile distribution status. Authentication parameters are required only if this information is not provided separately.

```
http[s]://<sms_server>/ipsProfileMgmt/distributionStatus?
<distribution_id>
```

Profile distribution XML schema

The Remote Profile Management API uses the following XML schema for profile distribution status requests.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:simpleType name="uuid">
 <xs:restriction base="xs:string">
  <xs:pattern value="[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]
    {4}-[0-9a-f]{4}-[0-9a-f]{12}"/>
  </xs:restriction>
 </xs:simpleType>

<xs:element name="distributions">
<xs:complexType>
 <xs:sequence>
  <xs:element name="distribution" minOccurs="1"
    maxOccurs="unbounded">
 <xs:complexType>
   <xs:sequence>
    <xs:element name="device" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:choice>
       <xs:element name="name" type="xs:string"/>
       <xs:element name="id" type="uuid"/>
       <xs:element name="shortID" type="xs:positiveInteger"/>
       <xs:element name="ipAddress" type="xs:string"/>
     </xs:choice>
    </xs:complexType>
  </xs:element>
 </xs:sequence>
```

```
<xs:attribute name="id" type="uuid"/>
 </xs:complexType>
  </xs:element>
   </xs:sequence>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

XML elements

The following table describes XML elements in the profile distribution request.

Element	Value	Definition
`distribution/id`	String	Internal ID assigned to the distribution session expressed in UUID format
`device/id`	String	Internal ID assigned to the device expressed in UUID format
`device/shortID`	Positive integer	Internal number assigned to the device
`device/name`	String	Name of the device
`device/ipAddress`	String	IP address string of the device

Retrieve Digital Vaccine information

The Remote Profile Management API allows you to view information the active Digital Vaccine (DV) on the SMS and all DVs stored on the SMS.

The following example shows the URL pattern for an API call to initiate a profile distribution. Authentication parameters are required only if this information is not provided separately.

`http[s]://<sms_server>/ipsProfileMgmt/dvInfo?request=[active|all]&smsuser=<user_name>&smspass=<password>`

Valid values for the `request` parameter are `active` (show the active Digital Vaccine on the SMS) and `all` (show a list of all Digital Vaccines on the SMS).

Show the active Digital Vaccine on the SMS

```
http[s]://<sms_server>/ipsProfileMgmt/dvInfo?
request=<active>&smsuser=<sms_user>&smspass=<password>
```

Show a list of all Digital Vaccines on the SMS

```
http[s]://<sms_server>/ipsProfileMgmt/dvInfo?
request=<all>&smsuser=<sms_user>&smspass=<password>
```

Current filter settings

Use the `getFilters` servlet to obtain information about current filter settings for specified filters in a specific profile. The Current Filter Settings Request is a POST request, and requires you to provide an XML file that identifies the profile and the filter(s).

The following example shows the URL format to request current filter settings.

```
http[s]://<sms_server>/ipsProfileMgmt/getFilters
```

When the SMS receives a Current Filter Settings service request, it performs the following functions:

1. Validates the filter ID using the DV metadata.
2. Finds the category the filter ID belongs to.
3. Finds the setting of the category from the profile specified by the Profile ID and version.
4. Sets the setting of the filter ID in the response XML.

Note: The setting of a given filter might be changed by IPS administrators. The changes are defined in the POLICY response XML defined by the existing service interface.

Current filter settings XML schema

The Remote Profile Management API uses the following XML schema for current filter settings status requests.

```
<?xml version="1.0" encoding="UTF-8"?>
 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="uuid">
   <xs:restriction base="xs:string">
    <xs:pattern value="[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]
     {4}-[0-9a-f]{4}-[0-9a-f]{12}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="getFilters">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="profile">
          <xs:complexType>
```

```
            <xs:attribute name="id" type="uuid"/>
            <xs:attribute name="name" type="xs:string"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="filter" maxOccurs="unbounded">
         <xs:complexType>
          <xs:sequence>
           <xs:element name="number" type="xs:positiveInteger"minOccurs="0"/>
           <xs:element name="name" type="xs:string" minOccurs="0"/>
           <xs:element name="signature-id" type="uuid" minOccurs="0"/>
           <xs:element name="policy-id" type="uuid" minOccurs="0"/>
          </xs:sequence>
         </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML elements

The following table describes XML elements in the current filter setting request.

Element	Value	Definition
`profile`	NULL element with these attributes: • id • name	IPS profile identified by ID expressed in UUID format and name string
`number`	Integer	Internally-assigned unique number for the filter
`name`	String	Name of the filter
`signature-id`	String	Internally-assigned ID to the filter expressed in UUID format
`policy-id`	string	Internally-assigned ID to the policy expressed in UUID format

The following sample shows an instance of filter current setting request XML:

```
<?xml version="1.0"?>
<getFilters>
    <profile name="Default"/>
    <filter>
        <number>3295</number>
    </filter>
    <filter>
        <signature-id>00000001-0001-0001-0001-000000000027</signature-id>
    </filter>
    <filter>
        <policy-id>00000002-0002-0002-0002-000000000051</policy-id>
    </filter>
    <filter>
        <name>0050: IP Options: Unknown Code</name>
    </filter>
</getFilters>
```

Current filter settings XML response

The current filter settings response is defined in the following XML schema format:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:simpleType name="uuid">
  <xs:restriction base="xs:string">
   <xs:pattern
value="[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}"/>
  </xs:restriction>
 </xs:simpleType>
  <xs:element name="filters">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="profile">
    <xs:complexType>
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute name="id" type="xs:string"/>
        <xs:attribute name="version" type="xs:string"/>
    </xs:complexType>
     </xs:element>
      <xs:element name="filter" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
              <xs:element name="policy-id" type="uuid"/>
              <xs:element name="version" type="xs:string"/>
              <xs:element name="locked" type="xs:boolean"/>
```

```
          <xs:element name="useParent" type="xs:boolean"/>
          <xs:element name="comment" type="xs:string" minOccurs="0"/>
          <xs:element name="description" type="xs:string" minOccurs="0"/>
          <xs:element name="severity" minOccurs="0">
           <xs:simpleType>
            <xs:restriction base="xs:string">
             <xs:enumeration value="Low"/>
             <xs:enumeration value="Minor"/>
             <xs:enumeration value="Major"/>
             <xs:enumeration value="Critical"/>
            </xs:restriction>
           </xs:simpleType>
         </xs:element>
            <xs:element name="enabled" type="xs:boolean"/>
            <xs:element name="actionset" minOccurs="0">
           <xs:complexType>
            <xs:attribute name="refid" type="uuid"/>
            <xs:attribute name="name" type="xs:string"/>
          </xs:complexType>
        </xs:element>
 <xs:element name="control">
  <xs:simpleType>
   <xs:restriction base="xs:string">
   <xs:enumeration value="Category"/>
   <xs:enumeration value="Filter"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
<xs:element name="afc" type="xs:boolean"/>
<xs:element name="policyGroup" minOccurs="0">
 <xs:complexType>
  <xs:attribute name="refid" type="uuid"/>
 </xs:complexType>
</xs:element>
<xs:element name="trigger" minOccurs="0">
 <xs:complexType>
  <xs:attribute name="threshold">
   <xs:simpleType>
    <xs:restriction base="xs:integer">
     <xs:minInclusive value="2"/>
     <xs:maxInclusive value="10000"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="timeout">
   <xs:simpleType>
    <xs:restriction base="xs:long">
     <xs:minInclusive value="0"/>
     <xs:maxInclusive value="999999"/>
```

```
          </xs:restriction>
         </xs:simpleType>
        </xs:attribute>
       </xs:complexType>
      </xs:element>
      <xs:element name="capability" minOccurs="0" maxOccurs="unbounded">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="enabled" type="xs:boolean"/>
         <xs:element name="actionset" minOccurs="0">
          <xs:complexType>
              <xs:attribute name="refid" type="uuid"/>
              <xs:attribute name="name" type="xs:string"/>
            </xs:complexType>
           </xs:element>
          </xs:sequence>
           <xs:attribute name="name" type="xs:string"/>
         </xs:complexType>
        </xs:element>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

XML elements

The following table describes the definitions of current filter setting response XML elements.

Element	Value	Definition
`profile`	NULL element with these attributes: • id • name • version	An IPS profile identified by an id expressed in UUID format, a name string and a version number string.
`name`	String	The name of the filter.

Element	Value	Definition
`policy-id`	String	An internally assigned id to the policy expressed in UUID format.
`version`	Integer	IPS TOS version that the filter is applicable.
`locked`	Boolean	Boolean variable indicating if the filter is locked. A locked filter cannot be remotely changed.
`useParent`	Boolean	Boolean variable indicating if the filter actionset setting is inherited from a parent profile.
`comment`	String	User comment on the filter.
`description`	String	Description of the filter.
`severity`	String taking one of these values: • Low • Minor • Major • Critical	Severity of the filter.
`enabled`	Boolean	Boolean variable indicating if the filter is enabled or disabled.
`actionset`	NULL element with these attributes: • refid • name	An actionset setting defined by a `refid` expressed in UUID format and a *name* string.
`control`	String taking one of these values:	Controlling element of the filter `actionset` setting.

Element	Value	Definition
	• Category • Filter	If the filter's `actionset` setting is controlled by its category action set setting, then the control will be "Category". If the filter's `actionset` setting is controlled by overriding its default action set setting, then the control will be "Filter".
`afc`	Boolean	Boolean variable indicating if the filter is managed by the IPS Adaptive Filter Configuration (AFC). If a filter is managed by AFC, then the filter will be automatically disabled when the IPS device is under heavy load and the given filter is being triggered without actual filter match.
`policyGroup`	NULL element with a *refid* attribute	An IPS profile group identified by a *refid* expressed in UUID format. The *policyGroup* element is never used by a filter.
`trigger`	NULL element with these attributes: • threshold • timeout	A filter's trigger frequency detection parameter. The threshold is used to specify the number of filter triggers. The timeout is used to specify the time period under which the number of triggers are being counted (in seconds). The trigger element is used only for scan/sweep filters.
`capability`	Element with a name attribute having these child elements: • enabled • actionset • refid	IPS device specific filter settings. The *name* attribute specifies the type of device. The *enabled* and *actionset* child elements specify the filter setting. These child elements have the same definition as those defined previously. The *refid* element maps to the action set ID for the capability.

The following sample shows an instance of filter current setting response XML:

```
<?xml version="1.0" encoding="utf-8"?>
 <filters xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="getFiltersResponse.xsd">
  <profile name="name" id="5d8814e0-8a58-11e1-23de-c4ae1e90eacf" version="1.1"/>
```

```
 <filter>
   <name>7120: TCP: Segment Overlap With Different Data, e.g., Fragroute</name>
    <policy-id>00000002-0002-0002-0002-000000007120</policy-id>
    <version>5400+</version>
    <locked>false</locked>
    <useParent>true</useParent>
    <severity>Low</severity>
    <enabled>false</enabled>
    <control>Category</control>
    <afc>true</afc>
     <capability name="n-series">
      <enabled>true</enabled>
 <actionset name="Block / Notify" refid="a6ae6a71-b685-49fb-a478-b558ea8ade2a"/>
     </capability>
  </filter>
 <filter>
   <name>3295: HTTP: ShoutCAST DNAS Format String Vulnerability</name>
   <policy-id>00000002-0002-0002-0002-000000003295</policy-id>
   <version>5200+</version>
   <locked>false</locked>
   <useParent>true</useParent>
   <severity>Critical</severity>
   <enabled>false</enabled>
   <control>Category</control>
   <afc>true</afc>
   <capability name="n-series">
     <enabled>true</enabled>
 <actionset name="Block / Notify" refid="a6ae6a71-b685-49fb-a478-b558ea8ade2a"/>
   </capability>
   <capability name="model-10">
    <enabled>true</enabled>
    <actionset name="Block / Notify" refid="a6ae6a71-b685-49fb-a478-b558ea8ade2a"/>
   </capability>
  </filter>
<filter>
  <name>0027: IP Options: Record Route (RR)</name>
  <policy-id>00000002-0002-0002-0002-000000000027</policy-id>
  <version>5200+</version>
  <locked>false</locked>
  <useParent>true</useParent>
  <comment>This is a comment</comment>
  <severity>Minor</severity>
  <enabled>true</enabled>
 <actionset refid="e0a0b14b-934c-11d6-93ca-0002b34b9580" name="Block"/>
  <control>Filter</control>
  <afc>true</afc>
 </filter>
<filter>
  <name>0051: IP: Source IP Address Spoofed (Impossible Packet)</name>
```

```
  <policy-id>00000002-0002-0002-0002-000000000051</policy-id>
  <version>5200+</version>
  <locked>false</locked>
  <useParent>true</useParent>
  <severity>Critical</severity>
  <enabled>true</enabled>
 <actionset refid="a6ae6a71-b685-49fb-a478-b558ea8ade2a" name="Block/Notify"/>
  <control>Category</control>
  <afc>true</afc>
 </filter>
<filter>
  <name>0050: IP Options: Unknown Code</name>
  <policy-id>00000002-0002-0002-0002-000000000050</policy-id>
  <version>5400+</version>
  <locked>false</locked>
  <useParent>true</useParent>
  <severity>Minor</severity>
  <enabled>false</enabled>
  <control>Category</control>
  <afc>true</afc>
 </filter>
</filters>
```

Update filter settings

The `setFilters` servlet allows you to apply policy changes to selected profiles. A Filter Setting Change Request uses the POST method to merge specified filter settings with a specific profile.

The following example shows the URL format for an Update Filter Settings request.

```
https://<sms_server>/ipsProfileMgmt/setFilters
```

Update filter settings XML schema

The Remote Profile Management API uses the following XML schema for filter settings change requests.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="uuid">
    <xs:restriction base="xs:string">
<xs:pattern value="[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="setFilters">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="profile">
          <xs:complexType>
```

```
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="id" type="uuid"/>
  </xs:complexType>
</xs:element>
<xs:element name="filter" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:choice>
        <xs:element name="policy-id" type="uuid"/>
        <xs:element name="signature-id" type="uuid"/>
        <xs:element name="number" type="xs:positiveInteger"/>
        <xs:element name="name" type="xs:string"/>
      </xs:choice>
      <xs:element name="locked" type="xs:boolean" minOccurs="0"/>
      <xs:element name="comment" type="xs:string" minOccurs="0"/>
      <xs:element name="control" minOccurs="0">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Category"/>
            <xs:enumeration value="Filter"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="actionset" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="refid" type="uuid"/>
          <xs:attribute name="name" type="xs:string"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="enabled" type="xs:boolean" minOccurs="0"/>
      <xs:element name="afc" type="xs:boolean" minOccurs="0"/>
      <xs:element name="useParent" type="xs:boolean" minOccurs="0"/>
      <xs:element name="trigger" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="threshold">
            <xs:simpleType>
              <xs:restriction base="xs:integer">
                <xs:minInclusive value="2"/>
                <xs:maxInclusive value="10000"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="timeout">
            <xs:simpleType>
              <xs:restriction base="xs:long">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="999999"/>
              </xs:restriction>
            </xs:simpleType>
```

```
                </xs:attribute>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

XML elements

The following table describes the XML elements in the filter settings change request that are used to identify or modify the filter settings.

Element	Value	Definition
`actionset`	NULL element with these attributes: • refid • name	Actionset setting defined by a refid expressed in UUID format or a name string
`afc`	Boolean	Boolean variable indicating if the filter is managed by the IPS Adaptive Filter Configuration (AFC) If a filter is managed by AFC, then the filter will be automatically disabled when the IPS device is under heavy load and the given filter is being triggered without actual filter match
`comment`	String	User comment on the filter
`control`	String taking one of these values: • category • filter	Controlling element of the filter's actionset setting. If an actionset is controlled by its category actionset, then the control is "Category" If an actionset is controlled by overriding its default actionset, then the control is "Filter"

Element	Value	Definition
`enabled`	Boolean	Boolean variable specifying if the filter should be enabled or disabled
`filter`	NULL element This value is read-only.	Parent element of the following elements
`locked`	Boolean	Boolean variable indicating if the filter is locked. A locked filter cannot be remotely changed
`number`	Integer This value is read-only.	Unique, internal assigned number for the filter
`name`	String This value is read-only.	Name of the filter
`policy-id`	String This value is read-only.	Internal ID assigned to the policy expressed in UUID format
`profile`	NULL element with two attributes: • id • name These values are read-only.	IPS profile identified by ID expressed in UUID format or name string
`signature-id`	String This value is read-only.	Internal ID assigned to the filter, expressed in UUID format
`trigger`	NULL element with these attributes: • threshold • timeout	A filter's trigger frequency detection parameter. The threshold is used to specify the number of filter triggers. The timeout is used to specify the time period under which the number of triggers are being counted (in seconds).

Element	Value	Definition
		The trigger element is used only for scan/sweep filters.
`useParent`	Boolean	Boolean variable indicating if a filter's actionset setting is inherited from a parent profile

The following sample shows an instance of update filter request XML:

```
<setFilters>
    <profile name="test"/>
    <filter>
        <number>7001</number>
        <actionset name="Block + Notify"/>
        <trigger threshold="10" timeout="5000"/>
    </filter>
    <filter>
        <number>3295</number>
        <actionset name="Block + Notify"/>
    </filter>
    <filter>
        <signature-id>00000001-0001-0001-0001-000000000027</signature-id>
        <enabled>false</enabled>
    </filter>
    <filter>
        <policy-id>00000002-0002-0002-0002-000000000051</policy-id>
        <comment>this is a comment</comment>
    </filter>
    <filter>
        <name>0050: IP Options: Unknown Code</name>
        <actionset refid="57ec4769-ca05-4dc5-8e79-a34c182adc48"/>
    </filter>
</setFilters>
```

Update filter settings XML response

For a sample XML response for the update filter settings, see *Current filter settings XML response* on page 61.

Remote Administration Management

The Remote Administration Management API enables you to perform remote SMS database backups and to retrieve SMS software version information. This API requires you to provide a valid username and password as part of the API call.

Parameters can be passed to the SMS either by entering a URL in a Web browser or by using a command line tool for HTTP scripting. When using http, passwords are sent in plain text. Using these APIs with HTTP should only be done on a trusted network. Passwords are not sent in plain text when using HTTPS.

Remote backups

The following table describes the remote backup parameters.

Parameter	Description
`type`	Destination type: `smb`, `nfs`, `scp`, `sftp`, `sms` (stored locally on the SMS—only one backup allowed at a time)
`location`	Destination path for backup file; does not apply for destination type `sms`
`username`	Type-specific username; used for destination types `smb`, `scp` and `sftp`
`password`	Type-specific password; used for destination types: `smb`, `scp` and `sftp`
`domain`	Type-specific domain; only used for destination type `smb`
`tos`	Number of most recent TOS packages to include (default value = 0)
`dv`	Number of most recent DV packages to include (default value = 1)
`events`	Include events data (boolean—default value `false`)

Parameter	Description
notify	Send email notification when backup has completed or failed (boolean—default value true)
timestamp	Use timestamp to build backup file name (boolean—default value true)
encryptionPass	Encrypt backup using supplied password (default null—do not encrypt)
smsuser	SMS username to use for the backup operation
smspass	SMS password

Back up locally to SMS (with defaults)

The following example shows the URL format you use when backing up locally to the SMS. The example omits optional parameters to accept default values.

```
http[s]://<sms_server>/smsAdmin/backup?type=sms
```

Back up with SCP (with some defaults)

The following example shows the URL format you use when creating a backup with SCP. The example specifies some parameters and omits others to accept default values.

```
http[s]://<sms_server>/smsAdmin/backup?
type=<scp>&location=<//203.0.113.0/home/usr/

backups/>&username=<scp_user>&password=<scp_pwd>&timestampName=<true>
```

Back up to SMS Server (with no defaults)

The following example shows the URL format to use when you backup to an SMB server, and specifies sample values for all parameters.

```
http[s]://<sms_server>/smsAdmin/backup?
type=<smb>&location=<//198.51.100.100/

backups/
sms.bak>&username=<smb_user>&password=<smb_pwd>&domain=<dom00>&tos=<1>
&dv=<1>&events=<false>&notify=<false>&timestampName=<true>
```

SMS version

The Remote Administration Management API allows you to use an HTTP GET method to request the SMS server software version.

The following example shows the URL format for the GET request. If the request is successful, the SMS returns a version number.

```
http[s]://<sms_server>/smsAdmin/info?
request=version&smsuser=<sms_user>

&smspass=<password>
```

Reputation Management

The Reputation Management API enables you to manage entries in the SMS Reputation Database by importing, creating, deleting, and resetting reputation entries.

Parameters can be passed to the SMS either by entering a URL in a Web browser or by using a command line tool for HTTP scripting. As part of the URL, passwords are sent in plain text; these APIs should only be used on a trusted network.

Import reputation entries

Importing reputation entries is a similar operation to the one referenced in *Import a profile* on page 49. The only parameter is `type`. The `type` parameter is optional and has a default value of `IPv4`. The possible values include the following:

- IPv4
- IPv6
- DNS

Reputation import rules

The SMS expects reputation files to be in CSV format and to follow the rules described in the following table. For reference, the rules are divided into files, fields, addresses and tags.

Item	Description
Files	
CSV format	The import file must be in comma-separated value (CSV) format
	Each line is made up of one or more fields separated by commas
	Each line represents one entry, and entries must not span lines
	Any line that has a first non-white space character of "#" is considered a comment. comment lines are discarded during import. There is no support for inline comments.

Item	Description
Blank lines	Import file may not contain any blank lines within the body
	Blank lines after the last line are ignored
Fields	
Double Quotes	A field may be enclosed in double-quotes For a value that contains a comma that is not a field separator, enclose the field in a double quote To represent a double-quote character within a quoted value, use two double-quotes
Addresses	
Address Types	Only one type of address (IPv4, IPv6 or DNS domain name) can be contained in the file. Mixing of types within a file is not allowed. The first field on each line must be the IPv4 address, IPv6 address or DNS name for that entry. The remaining fields on a line are optional. If present, remaining fields are processed as tag category/tag value pairs.
DNS Entries	A DNS entry matches any lookups that contain the specified string. For example, foo.com matches foo.com, www.foo.com, and images.foo.com. To specify an exact DNS entry match, enclose the DNS name in square brackets. For example, [foo.com] matches only foo.com, and does NOT match www.foo.com or images.foo.com.
CIDR Values	CIDR values are normalized. Any bits outside the portion of the address specified by the prefix length are changed to zero.
Tags	
SMS Parity	Any tag categories that appear in the file must exist on the SMS prior to import

Item	Description
Character Case	In tag category names and tag values, character case is significant
Yes/No tag categories	For yes/no tag categories, character case is insignificant
	For yes/no tag categories, the text "yes," regardless of case, denotes a yes value. All other values are considered no
Tag Pairs	Empty tag pairs (tag category/tag value) in fields are ignored If a tag category field is empty, an error occurs and the entry is not imported If a tag value field is empty, the corresponding tag category is discarded and the next field of the entry is processed. It is equivalent to the tag category not appearing on that line at all
	Tag pairs (tag category/tag value) do not have to appear in the same order on each line
Tag Categories	It is not necessary that every entry specify every tag category, or even the same tag categories as other entries in the file

Importing reputation entries

Importing reputation entries using the Web API works in a similar manner as the same function in the SMS client. If the reputation file does *not* have tags, the imported data merges with existing address values. If the file *does* have tags, the imported data overwrites existing address values.

Note: cURL is the preferred method to transfer reputation entries to the server.

The following example uses cURL to import a profile to the SMS. Authentication parameters are required only if this information is not provided separately.

```
curl -v -k -F "file=@/home/userid/Rep.txt"
```

```
"http[s]://<sms_server>/repEntries/import?
smsuser=<user_name>&smspass=<password>--&type=ipv4"
```

`-v` Enables "verbose" mode for debugging.

`-k` Allows cURL to perform insecure transfers over SSL connections.

`-F` Enables cURL to emulate a user pressing a submit button, using the Content-Type multipart/form-data according to RFC 2388.

Examples

The examples assume that the following tag categories are defined.

- Country (List)
- Approved (Yes/No)
- Comment (Text)

For the Country tag category, the following countries are defined:

- China
- Mexico
- United States

Examples are provided for the following areas:

- *Examples: file rules* on page 78
- *Examples: field rules* on page 79
- *Examples: address rules* on page 80
- *Examples: tags* on page 81

Examples: file rules

The import file must be in comma separated value (CSV) format. Each line is made up of one or more fields separated by commas.

RIGHT

```
1.2.3.0/24,Country,United States,Approved,yes
2.3.0.0/16,Country,Mexico,Approved,no
3.4.5.0/24,Country,China,Approved,yes
```

WRONG

```
1.2.3.0/24|Country|United States|Approved|yes
2.3.0.0/16|Country|Mexico|Approved|no
3.4.5.0/24|Country|China|Approved|yes
```

Examples: field rules

A field may be enclosed in double-quotes. This is mandatory when a value contains a comma that should not be treated as a field separator.

RIGHT

```
1.2.3.0/24,Country,United States,Approved,yes,Comment,"This comment,
contains a comma"

2.3.0.0/16,Country,Mexico,Approved,no

3.4.5.0/24,Country,China,Approved,yes
```

WRONG

```
1.2.3.0/24,Country,United States,Approved,yes,Comment,This comment,
contains a comma

2.3.0.0/16,Country,Mexico,Approved,no

3.4.5.0/24,Country,China,Approved,yes
```

Use two double-quotes to represent a double-quote character within a quoted value.

RIGHT

```
1.2.3.0/24,Country,United States,Approved,yes,Comment,"This comment
""contains"" quotes"

3.4.5.0/24,Country,China,Approved,yes
```

WRONG

```
1.2.3.0/24,Country,United States,Approved,yes,Comment,"This comment
"contains" quotes"

2.3.0.0/16,Country,Mexico,Approved,no

3.4.5.0/24,Country,China,Approved,yes
```

Each line represents one entry, and entries must not span lines.

RIGHT

```
1.2.3.0/24,Country,United States,Approved,yes

2.3.0.0/16,Country,Mexico,Approved,no

3.4.5.0/24,Country,China,Approved,yes
```

WRONG

```
1.2.3.0/24,Country,United
States,Approved,yes,2.3.0.0/16,Country,Mexico,Approved,no

3.4.5.0/24,Country,China, Approved,yes
```

Examples: address rules

The file must contain entries of only one type:

- IPv4 addresses
- IPv6 addresses
- or DNS domain names

Mixing of types within a file is not allowed.

RIGHT

```
1.2.3.0/24,Country,United States,Approved,yes

2.3.0.0/16,Country,Mexico,Approved,no

3.4.5.0/24,Country,China,Approved,yes
```

WRONG

```
1.2.3.0/24,Country,United States,Approved,yes

2.3.0.0/16,Country,Mexico,Approved,no

fc01:a63:1::/64,Country,China,Approved,yes
```

The first field on each line must be the IPv4 address, IPv6 address, or DNS name for that entry.

RIGHT

```
foo.com,Country,United States,Approved,yes

bar.com,Country,Mexico,Approved,no

foo.org,Country,China,Approved,yes
```

WRONG

```
Country,United States,foo.com,Approved,yes

bar.com,Country,Mexico,Approved,no

foo.org,Country,China,Approved,yes
```

The remaining fields on a line are optional. If present, they are processed as tag category/tag value pairs. (The first field-second field of the line-is a tag category name. The next field is a tag value. The next field is a tag category name. The next field is a tag value. Etc.)

RIGHT

```
1.2.3.0/24,Country,United States,Approved,yes
```

```
2.3.0.0/16
```

```
3.4.5.0/24,,,,
```

A DNS entry matches any lookups that contain the specified string. That is, "foo.com" matches "foo.com", "www.foo.com", and "images.foo.com". To specify an exact match, enclose the DNS name in square brackets. For example, "[foo.com]" matches only "foo.com", and not "www.foo.com" or "images.foo.com".

CIDR values are normalized. That is, any bits outside the portion of the address specified by the prefix length are changed to zero. For example, 192.168.66.127/24 are stored as 192.168.66.0/24.

Examples: tags

Any tag categories that appear in the file must exist on the SMS.

RIGHT

```
1.2.3.0/24,Country,United States,Approved,yes
```

```
2.3.0.0/16,Country,Mexico,Approved,no
```

```
3.4.5.0/24,Country,China,Approved,yes
```

WRONG

```
1.2.3.0/24,Country,United States,Approved,yes,Description,This tag
category is not defined2.3.0.0/16,Country,Mexico,Approved,no
```

```
3.4.5.0/24,Country,China,Approved,yes
```

Except for yes/no tag categories, character case is significant in all tag category names and tag values.

RIGHT

```
1.2.3.0/24,Country,United States,Approved,yes
```

```
2.3.0.0/16,Country,Mexico,Approved,no
```

```
3.4.5.0/24,Country,China,Approved,yes
```

WRONG

```
1.2.3.0/24,COUNTRY,United States,Approved,yes
```

```
2.3.0.0/16,country,Mexico,Approved,no
```

```
3.4.5.0/24,Country,CHINA,Approved,yes
```

For yes/no tag categories, the text "yes", regardless of case, denotes a yes value. All other values are considered no.

RIGHT

```
1.2.3.0/24,Country,United States,Approved,Yes
```

```
2.3.0.0/16,Country,Mexico,Approved,yes
```

```
3.4.5.0/24,Country,China,Approved,YES
```

WRONG

```
1.2.3.0/24,COUNTRY,United States,Approved,Yes
```

```
2.3.0.0/16,country,Mexico,Approved,On
```

```
3.4.5.0/24,Country,CHINA,Approved,T
```

Empty pairs of fields are ignored. If a tag category field is empty, an error occurs and the entry is not imported. If a tag value field is empty, the corresponding tag category is discarded and the next field of the entry is processed. It is equivalent to the tag category not appearing on that line at all.

RIGHT

```
1.2.3.0/24,,,Approved,Yes
```

```
2.3.0.0/16,,,,
```

```
3.4.5.0/24,Country,,Approved,
```

WRONG

```
1.2.3.0/24,,United States,Approved,yes
```

```
2.3.0.0/16,Country,Mexico,,yes
```

```
3.4.5.0/24,Country,China,Approved,yes
```

Tag category/tag value pairs do not have to appear in the same order on each line. It is not necessary that every entry specify every tag category, or even the same tag categories as other entries in the file. The only requirement is that tag categories must exist on the SMS prior to the import.

RIGHT

```
1.2.3.0/24,Country,United States,Approved,yes
```

```
2.3.0.0/16,Approved,no,Country,Mexico
```

```
3.4.5.0/24
```

Create reputation entries

The Reputation Management API enables you to add untagged reputation entries to the SMS reputation database and supports IPv4, IPv6, and DNS entries. The following examples demonstrate the URL format

to use to add reputation entries. Authentication parameters are required only if this information is not provided separately.

IPv4

```
http[s]://<sms_server>/repEntries/add?
smsuser=<user_name>&smspass=<password>&ip=<198.51.100.0>
```

IPv6

```
http[s]://<sms_server>/repEntries/add?
smsuser=<user_name>&smspass=<password>&ip=<ff01:0001:1461:feed::0002>
```

DNS

```
http[s]://<sms_server>/repEntries/add?
smsuser=<user_name>&smspass=<password>&dns=<some-url.com>
```

The SMS Reputation Management API allows you to add reputation entries to or delete reputation entries from the reputation database using HTTP calls.

When you add reputation entries, use a comma (,) as a delimiter between a tag category name and the tag category value.

Entries that are associated with a LIST tag category can include multiple values only when the **Allow Multiple Values?** check box is selected (on the Edit Tag Category dialog box in the SMS). The list values must be separated by ~~~. For example:

```
MalwareIpType,malwareSource~~~cncHost
```

If the LIST Tag Value Category is restricted to just one value, then the tag category name would be followed by the one value. For example: `MalwareIpType,infectedHost`

Adding a reputation entry requires a parameter called `TagData`. *Deleting* a reputation entry requires a parameter for `ip` or `dns` and the parameter `criteria` with the value `entry`.

Adding a reputation entry

The parameter values for `TagData` must be UTF-8 encoded.

The following example adds a reputation entry for tag category `MalwareIpType` and `CreatedDate`:

```
http[s]://<sms_server>/repEntries/add?
smsuser=<user_name>&smspass=<password>

&ip=1.1.1.1&TagData=MalwareIpType,infectedHost,CreatedDate,"Jan 22, 2014"
```

In the example above:

- `<sms_server>` is the IP address of the SMS server

- `1.1.1.1` is the IP address of the infected host (list value for `MalwareIpType` tag category)
- `"Jan 22, 2014"` is the list value for the `CreatedDate` tag category in the format MM DD, YYYY

Delete reputation entries

The Delete API can delete one or more IP addresses and DNS entries in a single call. Add one or more IP or DNS parameters to the request. For example:

```
http[s]://<sms_server>/repEntries/delete?
smsuser=<user_name>&smspass=<password>

&ip=1.1.1.1&ip=1.1.1.2&dns=malware.source1.com&

dns=malware.source2.com&criteria=entry
```

In above example:

- `<sms_server>` is the IP address of the SMS server
- `1.1.1.1` and `1.1.1.2` are the entries that will be deleted from the reputation database
- `malware.source1.com` and `malware.source2.com` are the DNS entries that will be deleted from the reputation database

Reset reputation entries

The Reputation Management API allows you to reset RepDV entries to their original state.

The following example shows the URL format to use to reset RepDV entries. Authentication parameters are required only if this information is not provided separately.

Reset RepDV

```
http[s]://<sms_server>/repEntries/delete?
smsuser=<user_name>&smspass=<password> &criteria=repdv
```

Query reputation entries

SMS Reputation Management API allows a client to query reputation user-provided tagged entries for an IP address or a DNS address. The following table describes the parameters to be specified in a query.

Parameter	Description
smsuser	Required. User name for an SMS administrator.
smspass	Required. Password for the user specified in the smsuser parameter.
ip	IP address of either IPv4 or IPv6 format. More than one IP address can be added to the query. At least one ip or dns parameter must be specified.
dns	DNS name. More than one DNS name can be added to the query. At least one ip or dns parameter must be specified.

The following example shows the host URL the client uses for the query:

```
http[s]://<sms_server>/repEntries/query
```

<sms_server> is the IP address of the SMS server

Successful HTTP response codes are 200 and 204. A response code of 200 indicates the query was a success and data is being returned. A response code of 204 indicates the query did not find the IP or DNS entries in the Reputation Database, and no data is being returned.

Note that at least one IP address or DNS name must be specified in a query request, but the request can contain only one type: *either* ip or dns. Below are some examples.

Example (IP parameters):

```
http[s]://<sms_server>/repEntries/query?
smsuser=<smsusername>&smspass=<smspassword>&ip=1.1.1.1&ip=1.1.1.2&ip=1.1.1.3
```

Example (DNS parameters):

```
http[s]://<sms_server>/repEntries/query?
smsuser=<smsusername>&smspass=<smspassword>
&dns=www.test1.com&dns=www.test2.com&dns=www.test3.com
```

The SMS responds to a query request with a mime-type of *text/plain* with default character encoding of UTF-8. For each IP address and DNS name specified in the query, the SMS will return all matching entries. Each value is terminated by a <new-line> character, so each returned value appears on its own line.

If you have a query IP address of 1.0.0.1 and the database has user entries of 1.0.0.1, 1.0.0.0/24, and 1.0.0.0/16 the returned result for this one IP will be 3 rows:

```
1.0.0.0/16,AtaHost,myata.device.com,MalwareIpType,infectedHost
```

```
1.0.0.0/24,AtaHost,myata.device.com,MalwareIpType,infectedHost

1.0.0.1,AtaHost,myata.device.com,MalwareIpType,infectedHost
```

As shown in the above example, the matching IP address or DNS is appended to the beginning of the reputation entries so you can see the entry that is stored in the database.

The format of the returned values is equivalent to the format used during the "Add" service:

```
<IP>, "Tag Class","Value",…

<DNS>, "Tag Class","Value",…
```

In the above example, note the following:

- `<IP>` is the IP address or subnet that matches the provided IP value
- `<DNS>` is the DNS name that matches the provided DNS value
- `Value` is either a single value or a list of values

If a Tag class supports multiple values, it is represented by a list of values separated with three tildes, for example: "value" and "value1~~~value2~~~value3".

The following examples show a sample request and a sample response.

Request:

```
http[s]://<sms_server>/ repEntries/query?
smsuser=<smsusername>&smspass=<smspassword> &ip=1.1.1.1&ip=1.1.1.2
```

Response:

```
1.1.1.1,AtaHost,myata.device.com,MalwareIpType,infectedHost1.1.1.2,

AtaHost,myata.device.com,ThreatScore,28,MalwareIpType,
cncHost~~~infectedHost,Fri Jan 31 00:00:00CST2014
```

Performance guidelines

Performance levels can vary greatly according to the number of files to be imported into the reputation database, the number of entries in each CSV file, the number and type of devices the SMS manages on the customer's network, and the customer's environment in general.

The SMS is limited to a maximum of 20 tag categories for the reputation database, and the maximum number of reputation entries is 6,000,000; however, we suggest limiting the number of entries in a file to 10,000. The maximum number of IP or DNS values that can be specified in a single query is 10,000.

The SMS can upload one CSV file at a time, and each file can contain multiple entries. Returned addresses are ordered from lowest to highest address, regardless of the order in which they are specified in a query.

Note: Flooding the SMS with multiple files (with single or few entries) can cause performance issues. We recommend monitoring the growth of the device distribution queue size to ensure that the devices can sustain the number of changes to the entries.

Note: In releases prior to SMS 4.1, the SMS prevented concurrent processing of uploaded files by refusing file uploads until the previous upload was complete. As such, we cannot guarantee that frequent updates will be processed, and data loss may still occur even when a successful HTTP response code of 200 is returned. While integration is possible in versions prior to 3.6, we recommend SMS 4.1 for production deployments as SMS version 4.1 supports concurrent file import.

The file upload interval on the advanced threat device should be configurable, as it will allow the network administrator to control the number of distributions that occur. Customers that have a large number of devices may want to increase the interval so that their SMS is not overloaded with frequent distributions.

As a best practice, customers must tune their deployment so that the rate of Advanced Threat Entry submissions to the SMS does not result in increasing the depth of the device distribution queue. The rate of submission depends on the following:

- The API used for the submission: CSV file import or ADD API
- The number of entries on the SMS. As such, a small reputation database has a higher number of distributions than a large reputation database
- The time it takes to synch reputation entries to the devices in the customer's deployment

Initial rate guidelines

The initial out-of-the-box guidelines for the CSV file import and Add/Delete API include the following:

- **CSV file import:** Limit the rate of submission to approximately no more than one CSV file every two minutes
- **Add/Delete API:** Bursts up to 1,000 in intervals of five minutes

CSV Add

Worst case: one entry per request

Submitting a file with only one entry (every few seconds or more) is the least efficient way to submit an entry to the SMS, as each request results in a distribution and a sync time, and the SMS can quickly get overwhelmed with distributions.

Note: The File Upload API uses the ADD API when a file contains 10 or less records, and this can help performance by reducing the number of distributions. This threshold can be configured with the assistance of the TippingPoint support team.

Best case: more than 1,000 entries per request

Batching a large number entries in one file is the most efficient way to add entries to the SMS, as there is no limit on the number of entries in a file, and each file will cause a distribution to the device.

Add API

One entry per request by design

A best practice when using this API is to send requests in bursts up to 1,000 entries, and bursts should be done in intervals to allow distributions to complete.

This depends on the number of entries in the reputation database and the time required to sync reputation entries to the devices in the customer's deployment. Sending one entry every few seconds is the worst case, as this results in a distribution, and the sync time between the SMS and the device can overwhelm the SMS with distributions.

Delete API

One entry per request by design

A best practice when using this API is to send requests in bursts up to 1,000 entries, and bursts should be done in intervals to allow distributions to complete.

This depends on the number of entries in the reputation database and the time required to sync reputation entries to the devices in the customer's deployment. Sending one entry every few seconds is the worst case, as this result in a distribution, and the sync time between the SMS and the device can overwhelm the SMS with distributions.

API minimum supported SMS versions

Upcoming SMS releases will add support for API features as indicated in the following table.

Release	API feature	Description
SMS 3.6 and later	Create reputation entries	Provides the ability to create reputation entries by importing them into the reputation database via CSV file.
SMS 4.1	Add reputation entries	Provides the ability to add reputation entries to the reputation database using HTTP.
SMS 4.1	Delete reputation entries	Provides the ability to delete reputation entries from the reputation database using HTTP.

Release	API feature	Description
SMS 4.1	Query reputation entries	Provides the ability to query reputation entries using HTTP.

Virtual Segment Management

Virtual segments can be set up to define traffic using a VLAN ID, an endpoint pair (source and/or destination IP addresses of a packet), or both. One or more physical segments are then assigned to the virtual segment. Virtual segments are members of a segment group, and the assigned devices are not exposed in segment group membership.

The Virtual Segment Management API enables you to create, update, and delete virtual segments through an HTTP interface instead of the SMS client. In addition, you can retrieve a list of virtual segments for SMS managed devices.

The XML schema for the Virtual Segment Management and response and sample XML are documented.

Special notes

- Virtual segments can be created that do not initially contain any physical segments.
- IPS devices with virtual segments that were configured locally on an IPS device and then added to the SMS are merged to the global virtual segment listing.
- A virtual segment must contain at least one VLAN ID, source IP, or destination IP traffic definition.

Note: A *named resource* is an individual resource, typically created to be included in a named resource group. You cannot create a named resource using the SMS Web API. Any named resources that appear in the file must exist on the SMS.

Create virtual segments

The Virtual Segment Management API enables you to add virtual segments to the SMS database. The following example shows the URI pattern for an API call to add a virtual segment. Authentication parameters are required only if this information is not provided separately.

```
curl -v -k -F "file=@NamedResourceExample.xml"

"http[s]://<sms_server>/virtualsegment/create?
smsuser=<user_name>&smspass=<password>"
```

Parameters

The only parameter to create a virtual segment is `file`. The `file` parameter is the name of the file that contains the virtual segment XML, and the virtual segment to be created on the device and on the SMS.

Examples

Create a virtual segment (any VLAN, any destination, and a specific source IP address)

```
    <virtualSegment>
```

```
        <name>AnyExample</name>
        <description></description>
        <virtualSegPosition positionType="FIRST">
        </virtualSegPosition>
        <sourceAddressList>
            <cidrList>
                <cidr>1.1.1.133</cidr>
            </cidrList>
        </sourceAddressList>
        <segmentGroup>
            <segmentGroupID>
                <name>Default</name>
            </segmentGroupID>
        </segmentGroup>
 </virtualSegment>
```

Create a virtual segment (named resource):

For the sample XML that you can use to adjust the virtual segment position, see *Update virtual segments* on page 92.

Note: Any named resources that appear in the file must exist on the SMS.

```
<virtualSegment>
        <name>NamedResourceExample</name>
        <description></description>
        <virtualSegPosition positionType="LAST">
        </virtualSegPosition>
        <vlanIdList>
          <namedVlanGroup>WAN-Group</namedVlanGroup>
        </vlanIdList>
        <sourceAddressList>
          <namedAddrGroup>AccountingDeptsrcAddress</namedAddrGroup>
        </sourceAddressList>
        <destinationAddressList>
          <namedAddrGroup>DMZ</namedAddrGroup>
        </destinationAddressList>
        <segmentGroup>
            <segmentGroupID>
                <name>Default</name>
            </segmentGroupID>
        </segmentGroup>
        <physicalSegments>
            <physicalSegment>
                <device>
                    <name>IPS_device_name</name>
                </device>
                <segmentNameList>
                    <segmentNames>Segment 1-1 (A &gt; B)</segmentNames>
```

```
                    <segmentNames>Segment 1-1 (A &lt; B)</segmentNames>
                    <segmentNames>Segment 1-2 (A &gt; B)</segmentNames>
                    <segmentNames>Segment 1-2 (A &lt; B)</segmentNames>
                </segmentNameList>
            </physicalSegment>
        </physicalSegments>
</virtualSegment>
```

Update virtual segments

The Virtual Segment Management API enables you to update the virtual segments in the SMS database. The following example shows the URI pattern for an API call to update a virtual segment. Authentication parameters are required only if this information is not provided separately.

```
curl -v -k -F "file=@updateAddDeviceSegment.xml"
```

```
"http[s]://<sms_server>/virtualsegment/update?
smsuser=<user_name>&smspass=<password>&vs=NamedResourceExample"
```

Parameters

Updating virtual segment entries is a simular operation to the one reference in *Create virtual segments* on page 90. Besides the `file` parameter, the only additional parameter is `vs`. The `vs` parameter is the name of the virtual segment to be updated on the device and on the SMS.

Example

Update a virtual segment (named resource):

The following sample XML shows how you can update a virtual segment by adjusting the virtual segment position.

For the sample XML that you can use to create a virtual segment, see *Create virtual segments* on page 90.

Note: Any named resources that appear in the file must exist on the SMS.

```
<virtualSegment>
       <name>NamedResourceExample</name>
       <description></description>
       <virtualSegPosition positionType="ORDINAL_POSITION">
          <ordinalPosition>3</ordinalPosition>
       </virtualSegPosition>
       <vlanIdList>
          <namedVlanGroup>WAN-Group</namedVlanGroup>
       </vlanIdList>
       <sourceAddressList>
          <namedAddrGroup>AccountingDeptsrcAddress</namedAddrGroup>
```

```
        </sourceAddressList>
        <destinationAddressList>
           <namedAddrGroup>DMZ</namedAddrGroup>
        </destinationAddressList>
        <segmentGroup>
            <segmentGroupID>
                <name>Default</name>
            </segmentGroupID>
        </segmentGroup>
        <physicalSegments>
            <physicalSegment>
                <device>
                    <name>IPS_device_name</name>
                </device>
                <segmentNameList>
                    <segmentNames>Segment 1-1 (A &gt; B)</segmentNames>
                    <segmentNames>Segment 1-1 (A &lt; B)</segmentNames>
                    <segmentNames>Segment 1-2 (A &gt; B)</segmentNames>
                    <segmentNames>Segment 1-2 (A &lt; B)</segmentNames>
                </segmentNameList>
            </physicalSegment>
        </physicalSegments>
    </virtualSegment>
```

Delete virtual segments

The Virtual Segment Management API enables you to delete the virtual segments in the SMS database. The following example shows the URI pattern for an API call to delete a virtual segment. Authentication parameters are required only if this information is not provided separately.

```
curl -v -k "http[s]://<sms_server>/virtualsegment/delete?
smsuser=<user_name>&smspass=<password>&vs=NamedResourceExample"
```

Parameters

The only parameter to delete a virtual segment is `vs`. The `vs` parameter is the name of the virtual segment to be deleted from the device and on the SMS.

Virtual segment XML schema

The Virtual Segment Management API uses the following XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema" >
```

```
<xs:simpleType name="uuid">
 <xs:restriction base="xs:string">
  <xs:pattern value="[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]
   {4}-[0-9a-f]{4}-[0-9a-f]{12}"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="vs_name">
 <xs:restriction base="xs:string">
  <xs:maxLength value="127"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="vlan_Constraint">
 <xs:restriction base="xs:int">
  <xs:minInclusive value="0"/>
  <xs:maxInclusive value="4095"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="vs_description">
 <xs:restriction base="xs:string">
  <xs:maxLength value="250"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="positionType">
 <xs:restriction base="xs:string">
  <xs:annotation>
   <xs:documentation>Placement of the object in the list, first, last,
    or somewhere in between</xs:documentation>
  </xs:annotation>
   <xs:enumeration value="FIRST" />
   <xs:enumeration value="LAST" />
   <xs:enumeration value="ORDINAL_POSITION" />
 </xs:restriction>
</xs:simpleType>
<xs:complexType name="messageList">
 <xs:sequence>
  <xs:element type="xs:string" name="message"
   minOccurs="1"
   maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="deviceResult">
 <xs:all>
  <xs:element name="device" type="deviceType"/>
  <xs:element name="success" type="xs:boolean"/>
  <xs:element name="messages" type="messageList"
   minOccurs="0" maxOccurs="1"/>
```

```
 </xs:all>
</xs:complexType>
<xs:complexType name="deviceResultList">
 <xs:sequence>
  <xs:element type="deviceResult" name="deviceResult"
   minOccurs="1" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="rangeType">
 <xs:all>
  <xs:annotation>
   <xs:documentation>Range (i.e. 5 - 90)</xs:documentation>
   </xs:annotation>
    <xs:element type="vlan_Constraint" name="start"/>
    <xs:element type="vlan_Constraint" name="end"/>
 </xs:all>
</xs:complexType>


<xs:complexType name="idName">
 <xs:choice>
  <xs:element name="id"   type="xs:string"/>
  <xs:element name="name" type="xs:string"/>
 </xs:choice>
</xs:complexType>
<xs:complexType name="cidrListType">
 <xs:sequence>
  <xs:element type="xs:string" name="cidr" maxOccurs="unbounded">
   <xs:annotation>
    <xs:documentation>1 or more repetitions:1
     or more repetitions:</xs:documentation>
   </xs:annotation>
  </xs:element>
 </xs:sequence>
</xs:complexType>
<xs:element name="virtualSegment"  type="virtualSegmentType"
  nillable="false" />
 <xs:element name="virtualSegmentList" type="virtualSegmentListType"
  nillable="false"/>
 <xs:complexType name="segmentGroupType">
  <xs:sequence>
   <xs:element type="segmentGroupIDType" name="segmentGroupID"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="sourceAddressListType">
  <xs:choice>
   <xs:annotation>
    <xs:documentation>You have a CHOICE of the next
     2 items at this level</xs:documentation>
```

```
   </xs:annotation>
    <xs:element type="cidrListType" name="cidrList">
     </xs:element>
      <xs:element type="xs:string" name="namedAddrGroup">
     </xs:element>
  </xs:choice>
</xs:complexType>
<xs:complexType name="vlanIdListType">
 <xs:sequence>
  <xs:annotation>
   <xs:documentation>VLAN can either be a 1 named resource
    or a list of integer/ranges</xs:documentation>
  </xs:annotation>
  <xs:choice>
   <xs:element type="vlanListType" name="vlanList" >
   </xs:element>
   <xs:element type="xs:string" name="namedVlanGroup">
   </xs:element>
  </xs:choice>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="virtualSegmentType" >
  <xs:annotation>
   <xs:documentation>Definition of the virtual segment</xs:documentation>
   <xs:documentation>Any optional fields should be omitted,
    no empty elements</xs:documentation>
   <xs:documentation>Required: Name, segmentGroup, one,
    two or all of: [vlanIdList,sourceAddressList,
    destinationAddressList]</xs:documentation>
   <xs:documentation>Optional: description, and physicalSegments.
    If physicalSegments is not provided no devices will be updated with the
    virtual segment</xs:documentation>
  </xs:annotation>
  <xs:all>
   <xs:element type="vs_name" name="name" />
   <xs:element type="vs_description" name="description"
    nillable="false" minOccurs="0"/>
   <xs:element type="virtualSegPositionType" name="virtualSegPosition"/>
   <xs:element type="vlanIdListType" name="vlanIdList"
    nillable="false" minOccurs="0">
   </xs:element>
   <xs:element type="sourceAddressListType" name="sourceAddressList"
    nillable="false" minOccurs="0">
   </xs:element>
   <xs:element type="destinationAddressListType" name="destinationAddressList"
    nillable="false" minOccurs="0">
   </xs:element>
   <xs:element type="segmentGroupType" name="segmentGroup" />
   <xs:element type="physicalSegmentsType" name="physicalSegments"
```

```
    nillable="false" minOccurs="0">
   </xs:element>
  </xs:all>
 </xs:complexType>
 <xs:complexType name="virtualSegmentListType">
  <xs:sequence>
   <xs:element type="virtualSegmentType" name="virtualSegment"
    nillable="false" minOccurs="1" maxOccurs="unbounded">
   </xs:element>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="destinationAddressListType">
  <xs:choice>
   <xs:annotation>
    <xs:documentation>You have a CHOICE of the next
     2 items at this level</xs:documentation>
   </xs:annotation>
   <xs:element type="cidrListType" name="cidrList">
   </xs:element>
   <xs:element type="xs:string" name="namedAddrGroup">
   </xs:element>
  </xs:choice>
 </xs:complexType>
 <xs:complexType name="segmentGroupIDType">
  <xs:choice>
   <xs:annotation>
    <xs:documentation>You have a CHOICE of the next
     2 items at this level</xs:documentation>
   </xs:annotation>
   <xs:element type="xs:string" name="id">
   </xs:element>
   <xs:element type="xs:string" name="name"/>
  </xs:choice>
 </xs:complexType>
 <xs:complexType name="virtualSegPositionType">
  <xs:sequence>
   <xs:element  nillable="true" type="xs:positiveInteger"
    minOccurs="0" name="ordinalPosition">
   </xs:element>
  </xs:sequence>
  <xs:attribute type="positionType" name="positionType"/>
 </xs:complexType>
 <xs:complexType name="deviceType">
  <xs:choice>
   <xs:annotation>
    <xs:documentation>You have a CHOICE of the next
     4 items at this level</xs:documentation>
   </xs:annotation>
   <xs:element type="uuid" name="id"/>
```

```
   <xs:element type="xs:positiveInteger" name="shortID"/>
   <xs:element type="xs:string" name="name"/>
   <xs:element type="xs:string" name="ipAddress"/>
  </xs:choice>
 </xs:complexType>
 <xs:complexType name="segmentNameListType">
  <xs:sequence>
   <xs:element type="xs:string" name="segmentNames"
    minOccurs="1" maxOccurs="unbounded">
    <xs:annotation>
     <xs:documentation>1 or more device segment names</xs:documentation>
    </xs:annotation>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="vlanIdRangeType" >
  <xs:choice>
   <xs:element name="vlanID" type="vlan_Constraint"/>
   <xs:element name="vlanRange" type="rangeType"/>
  </xs:choice>
 </xs:complexType>

 <xs:complexType name="vlanListType" >
  <xs:sequence>
   <xs:element name="vlan" type="vlanIdRangeType"
    minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="physicalSegmentsType">
  <xs:sequence>
   <xs:annotation>
    <xs:documentation>1 or more repetitions:</xs:documentation>
   </xs:annotation>
   <xs:element type="deviceSegmentsType" name="physicalSegment"
    maxOccurs="unbounded">
   </xs:element>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="deviceSegmentsType">
  <xs:sequence>
   <xs:element type="deviceType" name="device"/>
   <xs:element type="segmentNameListType" name="segmentNameList"/>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Virtual Segment XML elements

The following table describes the elements in the Virtual Segment Management XML schema.

Element	Value	Definition
`name`	String	Name of the virtual segment
`description` (optional)	String	Description for the virtual segment
`virtualSegPosition`		Indicates where in the list virtual segment is placed. You define the priority order for virtual segment so that any overlapping definitions are resolved. Attempting to define an overlapping virtual segment on a device which does not allow it will produce an error.
`virtualSegPosition/ positionType`	ORDINAL_POSITION, FIRST, LAST	Attribute; must be one of the three values
`virtualSegPosition/ ordinalPosition`	Positive Integer	Must be provided when positionType is ORDINAL_POSITION
`vlanIdList` (optional)		Used to assign a list of VLAN IDs, and/or VLAN ranges or a named object referencing a named VLAN group
`vlanIdList/vlanList`		Used when assigning a list of VLAN IDs and/or VLAN ranges to the virtual segment

Element	Value	Definition
`vlanIdList/vlanList/vlan`		Single element for either a VLAN ID or VLAN range
`vlanIdList/vlanList/vlan/vlanID`	Integer (1 to 4094)	VLAN ID
`vlanIdList/vlanList/vlan/vlanID/vlanRange`		Element containing a VLAN range
`vlanIdList/vlanList/vlan/vlanID/vlanRange/start`	Integer (1 to 4094)	VLAN ID start of the range
`vlanIdList/vlanList/vlan/vlanID/vlanRange/end`	Integer (1 to 4094)	VLAN ID end of the range
`vlanIdList/namedVlanGroup`	String	Named VLAN group identifier
`sourceAddressList` (optional)		Used to assign a list of IP addresses and/or IP address blocks or a named object referencing a named address group for the source address
`sourceAddressList/cidrList`		Used when providing a list of IP addresses and/or IP address blocks
`sourceAddressList/cidrList/cidr`		IP address or IP address block
`sourceAddressList/namedAddrGroup`	String	Named address group identifier
`destinationAddressList` (optional)		Used to assign a list of IP addresses, and/or IP address blocks or a named object

Element	Value	Definition
		referencing a named address group for the destination address
`destinationAddressList/ cidrList`		Used when providing a list of IP addresses and/or IP address blocks
`destinationAddressList/ cidrList/cidr`		IP address or IP address block
`destinationAddressList/ namedAddrGroup`	String	Named address group identifier
`segmentGroup`		Used when assigning virtual segment to a segment group
`segmentGroup/segmentGroupID`		Identifier element for the segment group
`segmentGroup/ segmentGroupID/name`	String	Name of the segment group
`segmentGroup/ segmentGroupID/id`	String	ID of the segment group
`physicalSegments` (optional)		Used for assigning the virtual segment to one or more segments on one or more devices
`physicalSegments/ physicalSegment`		Identifies the device and the segments to assign the virtual segment to

Element	Value	Definition
`physicalSegments/ physicalSegment/device`		Identifies the device
`physicalSegments/ physicalSegment/device/uuid`	String	UUID of the device
`physicalSegments/ physicalSegment/device/ shortID`	Positive integer	Short ID of the device
`physicalSegments/ physicalSegment/device/name`	String	Name of the device
`physicalSegments/ physicalSegment/device/ ipAddress`	String	IP Address of the device
`physicalSegments/ physicalSegment/ segmentNameList`		Element containing a list of the segment names
`physicalSegments/ physicalSegment/ segmentNameList/ segmentNames`	String	Name of the segment

Retrieve a list of virtual segments

The Virtual Segment Management API enables you to retrieve a list of all of the virtual segments in the SMS database. The following example shows the URI pattern for an API call to retrieve a list of virtual segments. Authentication parameters are required only if this information is not provided separately.

```
curl -v -k "http[s]://<sms_server>/virtualsegment/get?
smsuser=<user_name>&smspass=<password>"
```

Note: Use this operation to retrieve a list of virtual segments in XML format. Use the following links to download the XML schema from the SMS: *https://<sms_ip_or_hostname>/xsds/VirtualSegment.xsd* or *https://<sms_ip_or_hostname>/xsds/sms/response/xsd*.

Important: In addition to the list of virtual segments, the device NAME from the DEVICE table is retrieved in this operation. It is the only record from the DEVICE table that is retrieved. For more information, see *DEVICE table* on page 14.

Response codes

When you use `GET` or `POST` methods to retrieve a list of virtual segments or perform a Virtual Segment Management operation, the Web API captures a response code. Refer to the following table for a description of the available Web API response codes and their corresponding HTTP response codes.

Web API response code	HTTP response code	Description
0	200	Successful completion
100	401	Authentication error
200	400	Missing parameter error
205	400	Operation error
300	400	Input XML file error
305	500	Output result file error
310	400	Validation error
320	400	Resource error
500	500	Unexpected error

Examples

Retrieve list of virtual segments

```
<smsResponse>
    <service>virtualsegment</service>
    <operation>get</operation>
    <resultCode>0</resultCode>
    <resultDetails>
        <virtualSegments>
            <virtualSegment>
                <name>NamedResourceExample</name>
                <description></description>
                <virtualSegPosition positionType="ORDINAL_POSITION">
                    <ordinalPosition>1</ordinalPosition>
                </virtualSegPosition>
                <vlanIdList>
                    <namedVlanGroup>WAN-Group</namedVlanGroup>
                </vlanIdList>
                <sourceAddressList>
                    <namedAddrGroup>AccountingDeptsrcAddress</namedAddrGroup>
                </sourceAddressList>
                <destinationAddressList>
                    <namedAddrGroup>DMZ</namedAddrGroup>
                </destinationAddressList>
                <segmentGroup>
                    <segmentGroupID>
                        <name>Default</name>
                    </segmentGroupID>
                </segmentGroup>
                <physicalSegments>
                    <physicalSegment>
                        <device>
                            <name>IPS_device_name</name>
                        </device>
                        <segmentNameList>
                          <segmentNames>Segment 1-1 (A &gt; B)</segmentNames>
                          <segmentNames>Segment 1-1 (A &lt; B)</segmentNames>
                          <segmentNames>Segment 1-2 (A &gt; B)</segmentNames>
                          <segmentNames>Segment 1-2 (A &lt; B)</segmentNames>
                        </segmentNameList>
                    </physicalSegment>
                </physicalSegments>
            </virtualSegment>
        </virtualSegments>
    </resultDetails>
</smsResponse>
```

Create virtual segment

```
<smsResponse>
    <service>virtualsegment</service>
    <operation>create</operation>
```

```
    <resultCode>0</resultCode>
    <resultDetails>
        <deviceResults>
            <deviceResult>
                <device>
                    <name>IPS_device_name</name>
                </device>
                <success>true</success>
            </deviceResult>
        </deviceResults>
    </resultDetails>
</smsResponse>
```

External database

The SMS supports the following database options:

- **External access** - direct access to the database
- **External replication** - remote replication of the database

The external database can be used for customized reporting. For custom reports, you can access the SMS database directly or replicate the SMS to your external server. If you require data that the SMS reports do not routinely provide, you can set up an SMS External Database with a reporting tool of your choice.

External Replication provides a copy of the database that can be edited, backed up, or used for offloading report functions.

Note: The data that you access remotely is read-only and cannot be changed.

External access

Setting up the Access service allows an external database tool to access data in the SMS. You must configure the SMS for external access before you configure your external application. You must reboot the SMS to enable or disable this service.

External replication

Setting up the replication service allows an external database server to replicate data from the SMS. You must reboot the SMS to enable or disable this service.

Configure the SMS for external access

This service opens a MariaDB read-only database for any third-party access or reporting tool. The read-only database name is **ExternalAccess**.

Note: Running complex report against SMS server may slow down the SMS response time significantly.

1. In the SMS, go to **Admin** > **Database**.
2. On the External Database Settings panel, click **Edit**.
3. In the Edit External Database Settings wizard, select **External Access Settings**.
4. Select **Enable external database access** to enable the service. (To disable the service, clear the check box.)
5. Provide the following:
    - **Username** – Provide the user name for an account with sufficient rights to read all the desired data from the SMS database.

- **Password** – Provide the password for the user account. Retype the password in the Confirm Password field.

6. If you changed the external access settings, click **Reboot** to restart the SMS server and initialize the service.

   Note: Follow your company's server downtime policies, including notification to SMS clients of a pending reboot. Before you reboot the SMS, gracefully stop other client connections to the server.

7. Click **OK**.

   If you verification fails or you encounter issues, check the following items:

   - Make sure that the username and password on the database are the same as the ones you set up on the SMS client.
   - Make sure to reboot the SMS before you try to access the database.

Configure the SMS for replication

This service allows an external database server to replicate data from the SMS. Using an external database for data replication allows you to offload report processing to an external server which can provide performance gains to your existing system. You must reboot the SMS to completely enable or disable this service.

Before you begin, make sure that your replication system has sufficient disk space to accommodate the database and any increase in size due to additional data or reporting.

1. In the SMS, go to **Admin** > **Database**.
2. On the External Database Settings panel, click **Edit**.
3. In the Edit External Database Settings wizard, select **External Replication Settings**.

   Note: To configure external database replication, you must create an SMS database snapshot, and then copy the snapshot to the target replication system and import it into a MariaDB database before the SMS server can replicate its data to the target system.

4. Select **Enable external database replication** to enable the service. (To disable the service, clear the check box.)
5. Provide the following:

   - **Username** – Provide the user name for an account with sufficient rights to read all the desired data from the SMS database.
   - **Password** – Provide the password for the user account. Retype the password in the Confirm Password field.

6. If you changed the replication settings, click **Reboot** to restart the SMS server and initialize the service.

Note: Follow your company's server downtime policies, including notification to SMS clients of a pending reboot. Before you reboot the SMS, gracefully stop other client connections to the server.

7. Click **Create Snapshot**, and select **Include Events in Snapshot** if you want the snapshot to include event data.

   Note: The snapshot is saved locally on the SMS server. You must copy the snapshot to the target replication system and import it into a new or existing MariaDB database before the SMS server can replicate its data to the target system.

8. Click **OK**.

   Note: External database replication and the SMS High Availability (HA) features both leverage the same functionality in the underlying MariaDB database. The SMS database does not support replication to multiple destinations; therefore, we do not recommend using SMS HA and external database replication at the same time.

Configure the SMS to enable restricted access

This service allows access to the external database to be restricted to a set of IP addresses.

1. In the SMS, go to **Admin** > **Database**.
2. On the External Database Settings panel, click **Edit**.
3. In the Edit External Database Settings wizard, select **Access Restrictions**.
4. Select **Enable restricted access** to enable the service. (To disable the service, clear the check box.)
5. Provide the following:
   - **Named IP Address Group** – To restrict a set of IP addresses, click the arrow, and either select a Named IP Address Group or create a new one.
6. Click **OK**.

ALERTS table - ExternalAccess

The database name for external access is **ExternalAccess**. With a few exceptions, the schema for the External Database Access ALERTS table is the same as Web API schema for the ALERT_TYPE table. See *ALERTS table* on page 29.

The following table includes the exceptions:

Column	Description
DEVICE_ID	**type change**. Unique identifier for the device in the format of integer, which is much faster when used in a query.
SRC_IP_ADDR_2	**New field**. Introduced for IPv6 support. Represents the higher 64 bit for the IPv6 source addresses. For IPv4 address, this field has a NULL value.
DST_IP_ADDR_2	**New field**. Introduced for IPv6 support. Represents the higher 64 bit for the IPv6 destination addresses. For IPv4 address, this field has a NULL value.

Replication: database schema

A list of tables created when you dump the snapshot file to the replicated database server. Some of the tables are for internal use only. The rest of tables are divided into two categories like Web Service API - Data Dictionary and Events Data. The tables are very similar with Web Server API with minor differences.

The following section detail the tables:

- *DataDictionary* on page 9
- *Events Data* on page 29

DataDictionary

See the following sections for more information on the tables in the database:

ACTIONSET table on page 12

CATEGORY table on page 13

NOTICE_ACTION table on page 14

POLICY table on page 15

POLICY_GROUP_LOOKUP table on page 15

PROFILE table on page 16

SEVERITY table on page 19

SIGNATURE table on page 19

Events Data

See the following sections for more information on the tables in the database:

Packet trace

The SMS Packet Trace feature compiles information about packets that have triggered a filter. Packet trace encapsulates the information according to requirements set for the filter in the SMS.

Packet trace options are configured for an action set, and an action set is specified for each filter. Filters are distributed to devices according to profiles. If a filter uses an action set for which packet trace logging is enabled, then you can view the compiled and stored packet trace information for events that triggered the filter.

The SMS saves packet trace information to a PCAP file. Two retrieval options are available for a packet trace:

- *Device-based packet trace* on page 111
- *Events-based packet trace* on page 111

Device-based packet trace

Device-based packet trace compiles PCAP information for a particular device from the SMS database. The following example shows the URI format to obtain a device-based packet trace. The `deviceId` in the example is the SHORT_ID for the device. For more information, see *DEVICE table* on page 14.

```
http[s]://<sms_server>/pcaps/getByDevice?deviceId=<SHORT_ID>
```

Events-based packet trace

To obtain all the PCAP information from the SMS for a group of events, you must know the event IDs.

Event IDs are included in data sent to a remote syslog server. For information about configuring and using Remote Syslog, see the *Security Management System User Guide*, and refer to the current SMS Deployment Note available from the TMC.

Setting up event-based packet trace

1. Set up a remote syslog server.
2. Add all the event IDs to a file as a comma separated list (new line breaks are also allowed).
3. Use cURL to upload the file to the Web server.

   The following example demonstrates a POST request to upload the file using cURL:

   ```
   curl -k -v -F "file=@<filepath/to/eventidfile.txt>"
   "https://<sms_server>/pcaps/getByEventIds?
   smsuser=<user_name>&smspass=<password>"
   ```

The result outputs to STDOUT and can be redirected to a file with a '>' operator.

MIB files for the SMS

A management information base (MIB) is a type of database that is used to manage devices in a communications network. Database entries are addressed through object identifiers (OIDs). MIB files are descriptions of network objects that can be managed using the Simple Network Management Protocol (SNMP). The format of the MIB is defined as part of the SNMP.

This information includes the following topics:

SMS MIBs

You can download TippingPoint SMS MIB files from the TMC at *https://tmc.tippingpoint.com*. On the TMC website, navigate to the Documentation area for this product release, and then select **SMS MIBS**.

The compressed file contains two MIB files:

- **TPT-SMSMIBS** defines monitoring functions
- **TPT-SMS-TRAP-MIB** defines the SMS traps

For more information about these MIBs, refer to the *TippingPoint Operating System MIB Guide*, available on the TMC.

Public MIB files

Publicly available UCD-SNMP-MIB and UCD-DISKIO-MIB definitions can be used to query SMS health values. These files can be downloaded from the following locations:

- *http://net-snmp.sourceforge.net/docs/mibs/*
- *http://net-snmp.sourceforge.net/docs/mibs/UCD-SNMP-MIB.txt*
- *http://net-snmp.sourceforge.net/docs/mibs/UCD-DISKIO-MIB.txt*

Note that only the SMS Health Section OIDs listed in *Health monitoring* on page 113 are supported.

Health monitoring

The following table lists the OIDs that are used to graph and display values in the SMS Health section of the SMS client.

Section	Description	OID
CPU	CPU_USER	1.3.6.1.4.1.2021.11.50.0
	CPU_SYS	1.3.6.1.4.1.2021.11.52.0
	CPU_IDLE	1.3.6.1.4.1.2021.11.53.0
Filesystem	FS_DSKPATH	1.3.6.1.4.1.2021.9.1.2
	FS_DEVPATH	1.3.6.1.4.1.2021.9.1.3
	FS_TOTAL	1.3.6.1.4.1.2021.9.1.6
	FS_AVAIL	1.3.6.1.4.1.2021.9.1.7
	FS_USED	1.3.6.1.4.1.2021.9.1.8
	FS_PERCENT	1.3.6.1.4.1.2021.9.1.9
	FS_IPERCENT	1.3.6.1.4.1.2021.9.1.10
High Availability	HA	1.3.6.1.4.1.2021.8.1.101.34
Memory	SWAP_TOTAL	1.3.6.1.4.1.2021.4.3.0
	SWAP_AVAIL	1.3.6.1.4.1.2021.4.4.0
	REALMEM_TOTAL	1.3.6.1.4.1.2021.4.5.0
	REALMEM_AVAIL	1.3.6.1.4.1.2021.4.6.0

Section	Description	OID
Network Traffic	ETHO_RX_BYTES	1.3.6.1.4.1.2021.8.1.101.1
	ETHO_RX_PACKETS	1.3.6.1.4.1.2021.8.1.101.2
	ETHO_RX_ERRORS	1.3.6.1.4.1.2021.8.1.101.3
	ETHO_RX_DROPPED	1.3.6.1.4.1.2021.8.1.101.4
	ETHO_RX_FIFO_ERRORS	1.3.6.1.4.1.2021.8.1.101.5
	ETHO_RX_FRAME_ERRORS	1.3.6.1.4.1.2021.8.1.101.6
	ETHO_RX_COMPRESSED	1.3.6.1.4.1.2021.8.1.101.7
	ETHO_TX_BYTES	1.3.6.1.4.1.2021.8.1.101.8
	ETHO_TX_PACKETS	1.3.6.1.4.1.2021.8.1.101.9
	ETHO_TX_ERRORS	1.3.6.1.4.1.2021.8.1.101.10
	ETHO_TX_DROPPED	1.3.6.1.4.1.2021.8.1.101.11
	ETHO_TX_FIFO_ERRORS	1.3.6.1.4.1.2021.8.1.101.12
	ETHO_TX_CARRIER_ERRORS	1.3.6.1.4.1.2021.8.1.101.13
	ETHO_TX_COMPRESSED	1.3.6.1.4.1.2021.8.1.101.14
	ETHO_MULTICAST	1.3.6.1.4.1.2021.8.1.101.15
	ETHO_COLLISIONS	1.3.6.1.4.1.2021.8.1.101.16

Section	Description	OID
	ETH1_RX_BYTES	1.3.6.1.4.1.2021.8.1.101.17
	ETH1_RX_PACKETS	1.3.6.1.4.1.2021.8.1.101.18
	ETH1_RX_ERRORS	1.3.6.1.4.1.2021.8.1.101.19
	ETH1_RX_DROPPED	1.3.6.1.4.1.2021.8.1.101.20
	ETH1_RX_FIFO_ERRORS	1.3.6.1.4.1.2021.8.1.101.21
	ETH1_RX_FRAME_ERRORS	1.3.6.1.4.1.2021.8.1.101.22
	ETH1_RX_COMPRESSED	1.3.6.1.4.1.2021.8.1.101.23
	ETH1_TX_BYTES	1.3.6.1.4.1.2021.8.1.101.24
	ETH1_TX_PACKETS	1.3.6.1.4.1.2021.8.1.101.25
	ETH1_TX_ERRORS	1.3.6.1.4.1.2021.8.1.101.26
	ETH1_TX_DROPPED	1.3.6.1.4.1.2021.8.1.101.27
	ETH1_TX_FIFO_ERRORS	1.3.6.1.4.1.2021.8.1.101.28
	ETH1_TX_CARRIER_ERRORS	1.3.6.1.4.1.2021.8.1.101.29
	ETH1_TX_COMPRESSED	1.3.6.1.4.1.2021.8.1.101.30
	ETH1_MULTICAST	1.3.6.1.4.1.2021.8.1.101.31
	ETH1_COLLISIONS	1.3.6.1.4.1.2021.8.1.101.32

Section	Description	OID
`Temperature`	`TEMPERATURE`	1.3.6.1.4.1.2021.8.1.101.33