TREND MICRO™

5.0.0 TippingPoint™ Security Management System (SMS)

Web API Guide

Legal and notice information

Security Management System Web API Guide

Contents

SMS Web API

The SMS Web API provides access to the following set of SMS features:

HTTPS or HTTP service to the SMS is required to send API requests to the SMS. By default, HTTPS service to the SMS is enabled. For more information, see "Server properties" in the *Security Management System User Guide*.

Authentication and authorization

The SMS supports the following authentication methods for the SMS web API:

- Query string parameters encoded in URL: `smsuser={username}&smspass={password}`
- HTTP Basic authentication: `-u {username}:{password}` option in cURL

Note: You can customize or replace the default SMS SSL X509 certificate in the SMS Admin workspace.

Web access on the SMS includes access to the SMS through the web client and web API requests. By default, authentication with a valid username and password is required for web access. For more information, see "System preferences" in the *Security Management System User Guide*.

We recommend that only users with the superuser role have web access for full authorization. For more information, see "Managing user roles" in the *Security Management System User Guide*.

Errors

The SMS web API returns one of the following HTTP status codes if the request is unsuccessful.

Code	Description
400	Bad request – Malformed parameter or request.
401	Unauthorized – Missing or incorrect credentials.
404	Not Found – Invalid or nonexistent requested source.
412	Preconditioned Fail – Unexpected error. Check the SMS system log for more information.
500	Internal Server Error – Server-side exception. Check the SMS system log for more information.

Profile management

The profile management API enables you to export, import, and distribute an SMS profile and to create and update filters. In addition, you can retrieve profile distribution status and data about the Digital Vaccine (DV) on the SMS.

For more information, see the *Security Management System User Guide*.

Shared settings

Profiles include shared settings, such as action 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. Instead, the SMS 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. Review services any time a profile is imported from a different user or from a different environment.

Export a profile

Use the `exportProfile` method to export a profile to the SMS Web "Exports" directory or to a fully specified SMB or NFS server location. Location parameters are required when you export to SMB or NFS remote directories.

Note: Profile packages typically remain unchanged. If you want to change the files within a profile package, update the md5sum in the `sms-security-manifest` file before importing the profile package back into the SMS.

Definition

```
ipsProfileMgmt/exportProfile
```

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

Parameter	Description
`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

Examples

The following example exports the MyProfile profile to the Export directory on the SMS server.

```
http[s]://<sms_server>/ipsProfileMgmt/exportProfile?profileName=MyProfile
&smsuser=<sms_user>&smspass=<password>
```

The following example exports the Default profile to an SMB server.

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

The following example exports the Default profile to an NFS server.

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

Import a profile

Use the `importProfile` method to import an exported profile package to the SMS. Check the version details section to see the name of the profile the current profile was imported from and the date of the update.

Note: If you change the profile, ensure that you maintain the same format. If you are importing a changed profile, ensure that you update the md5sum in the `sms-security-manifest` file in the profile package.

Definition

```
ipsProfileMgmt/importProfile
```

Parameters

Parameter	Description
`importAction`	Required. Possible values include the following: • `add`: Adds a completely new profile; must have an unused name or import fails. • `combine_add`: Adds new settings and merges non-conflicting changes into an existing profile. • `combine_change`: Adds new settings to and overwrites existing settings of an existing profile with settings of the new profile. • `replace`: Overwrites contents of SMS profile with those of the profile being imported; name and UUID remain the same; snapshot of replaced profile occurs and updated profile gets new version.
`targetProfileName`	The name of the existing profile in the SMS profile inventory. Required for all replace and combine actions. **Note:** The profile must exist in the SMS profile inventory. If the specified profile does not exist or is not specified in the request, the operation fails, an error is returned, and the audit log is updated with information. If the specified profile

Parameter	Description
	does exist, the specified `importAction` is performed, the target profile version is updated, and the audit log is updated with information.
`replacedProfileName`	The name of the imported profile that will have its contents applied to the existing profile in the SMS profile inventory. Required for all replace and combine actions. **Note:** The profile must be specified in the request. If the specified profile does not exist or is not specified in the request, the operation fails, an error is returned, and the audit log is updated with information.

Examples

Add a new profile:

```
curl -k -u <sms_user>:<password> -F "file=@</path/to/import.pkg>"
"https://<sms_server>/ipsProfileMgmt/importProfile?importAction=add"
```

Combine with an existing profile and add new settings:

```
curl -k -u <sms_user>:<password> -F "file=@</path/to/import.pkg>"
"https://<sms_server>/ipsProfileMgmt/importProfile?importAction=combine_add
&targetProfileName=<profile_name_on_sms>&replacedProfileName=
<adding_profile_name>"
```

Combine with an existing profile, adding new and overwriting existing settings:

```
curl -k -u <sms_user>:<password> -F "file=@</path/to/import.pkg>"
"https://<sms_server>/ipsProfileMgmt/importProfile?importAction=combine_change
&targetProfileName=<profile_name_on_sms>&replacedProfileName=
<adding_overwriting_profile_name>"
```

Replace the contents of an existing profile:

```
curl -k -u <sms_user>:<password> -F "file=@</path/to/import.pkg>"
"https://<sms_server>/ipsProfileMgmt/importProfile?importAction=replace
&targetProfileName=<profile_name_on_sms>&replacedProfileName=
<replacing_profile_name>"
```

Distribute a profile

Use the `distributeProfile` method to initiate a profile distribution to a single segment target or to a segment group.

Definition

```
ipsProfileMgmt/distributeProfile
```

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

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 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`	Empty 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

Element	Value	Definition
`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

Use the `distributionStatus` resource to determine the success or failure and the duration of a distribution session in a POST request. The actual percent-complete progress and predicted end-time are not available.

Definition

```
ipsProfileMgmt/distributionStatus
```

Parameters

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.

Example

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

Profile distribution XML schema

The 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

Element	Value	Definition
`device/name`	String	Name of the device
`device/ipAddress`	String	IP address string of the device

Create a traffic management filter

Use the `createTrafficMgmt` method to create a traffic management filter.

Definition

```
ipsProfileMgmt/createTrafficMgmt
```

Parameters

Note: Parameter names and enumerated values are not case sensitive.

The following parameters are required.

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 parameters 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`

Example

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

Retrieve current filter settings

Use the `getFilters` method to retrieve current filter settings for particular filters in a profile with an XML file with the profile and filter details.

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 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.

Example

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

Current filter settings XML schema

The 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`	Empty 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 filter ID expressed in UUID format
`policy-id`	string	Internally-assigned policy ID expressed in UUID format

The following sample shows an instance of a filter 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`	Empty 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.
`policy-id`	String	An internally-assigned policy ID 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	Severity of the filter.

Element	Value	Definition
	• Critical	
`enabled`	Boolean	Boolean variable indicating if the filter is enabled or disabled.
`actionset`	Empty 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: • Category • Filter	Controlling element of the filter `actionset` setting. If the filter's `actionset` setting is controlled by its category action set setting, then the control is "Category". If the filter's `actionset` setting is controlled by overriding its default action set setting, then the control is "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 is automatically disabled when the IPS device is under heavy load and the given filter is triggered without an actual filter match.
`policyGroup`	Empty 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`	Empty 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.

Element	Value	Definition
`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

Use the `setFilters` method to apply policy changes to selected profiles with an XML file with the profile and filter details.

Example

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

Update filter settings XML schema

The 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.

Element	Value	Definition
`actionset`	Empty 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	Controlling element of the filter's actionset setting. If an actionset is controlled by its category actionset, then the control is "Category"

Element	Value	Definition
	• filter	If an actionset is controlled by overriding its default actionset, then the control is "Filter"
`enabled`	Boolean	Boolean variable specifying if the filter should be enabled or disabled
`filter`	Empty 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`	Empty 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

Element	Value	Definition
`trigger`	Empty 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.
`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 16.

Retrieve Digital Vaccine information

Use the `dvInfo` resource to view the active DV on the SMS and all DVs stored on the SMS.

Definition

```
ipsProfileMgmt/dvInfo
```

Parameters

Parameter	Description
`request`	Required. Possible values include the following: • `active`: active DV on the SMS. • `all`: a list of all DVs on the SMS.

Examples

The following example retrieves the active DV on the SMS.

```
http[s]://<sms_server>/ipsProfileMgmt/dvInfo?request=active
```

The following example retrieves a list of all DVs on the SMS.

```
http[s]://<sms_server>/ipsProfileMgmt/dvInfo?request=all
```

Reputation Management

The Reputation Management API enables you to manage the SMS Reputation database by importing, adding, deleting, and querying Reputation entries.

For more information, see the *SMS User Guide* and the *URL Reputation Filtering Deployment and Best Practices Guide*.

Reputation management best practices

Monitor the device distribution queue to identify the appropriate time interval for submitting the Reputation Management API requests in your environment.

The following factors can affect performance levels:

- The method used for the Reputation entries submission – `import` or `add`. Use `import` with a large number of entries to reduce the number of distributions.

- The number of files to be imported into the Reputation database and the number of entries in each file.
- The number of entries on the SMS. A bigger reputation database takes longer to copy and distribute, resulting in less frequent distributions. For improved performance, limit the entries in the Reputation database to 6,000,000.
- The number and type of devices that the SMS manages. Newer models load the entries faster. If you have a large number of devices, increase the interval of entry submission so that the SMS is not overloaded with frequent distributions.

Syntax rules for import files

The import file must be in a comma-separated value (CSV) format with each line representing a Reputation entry without any blank lines. Comment lines are discarded during import. Each line is made up of one or more fields separated by commas.

URL Reputation entries

For URL entries, the import file must be delimited by pipe (|) instead of commas, and entries can be URLs only or URLs associated with one or more tags. Each line is made up of one or more fields separated by pipes. For more information about the URL import guidelines, see the *URL Reputation Filtering Deployment and Best Practices Guide*.

Construct your entries using the fields in the following table.

Field	Description	Required
Address	• The first field on each line must be the IPv4 address, IPv6 address, DNS name, or URL for that entry. The remaining fields on a line are optional. If present, remaining fields are processed as tag category/tag value pairs. • Only one type of address (IPv4, IPv6, DNS name, or URL) can be contained in a file. • 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`]. • CIDR values are normalized. Any bits outside the portion of the address specified by the prefix length are changed to zero. For example, 192.168.66.127/24 is stored as 192.168.66.0/24.	Yes

Field	Description	Required
Tag category/ tag value pairs	If the Reputation entry within the file does not have tags, the imported entry merges with the values of the existing entry. If the Reputation entry within the file does have tags, the imported entry merges and overwrites the values of the existing entry. • Any tag categories in the file must exist on the SMS prior to import. • Tag category/value pairs do not have to be listed in the same order on each line. The entries in the file do not have to list all the tag categories or specify the ones shared with other entries in the file. • 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; the net result is equivalent to the tag category not appearing on that line at all. • Except for yes/no tag categories, character case is significant in all tag category names and tag values. • For yes/no tag categories, the text `yes`, regardless of case, denotes a yes value. Any other text is considered a no value. • For list categories, the list values must be separated by `~~~`. • A field can be enclosed in double-quotes; this is mandatory when a value contains a comma that should not be treated as a field separator. • To represent a double-quote character within a quoted value, use two double-quotes.	No

Import file example

For the example in this section, the following tag categories are defined:

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

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

- China

- Mexico
- United States

The following example shows a file with IPv4 reputation entries.

```
1.2.3.0/24,Country,United States,Approved,yes
2.3.0.0/16,Country,Mexico,Approved,no
3.4.5.0/24,Approved,yes,Country,China
1.2.3.0/24,Country,United States,Approved,yes,Comment,"This
    comment, contains a comma"
1.2.3.0/24,Country,United States,Approved,yes,Comment,"This
    comment ""contains"" quotes"
2.3.0.0/16
3.4.5.0/24,,,,
```

Import Reputation entries

Use the `import` method to upload a file with one or more Reputation entries. The SMS can upload one file at a time, and each file can contain multiple entries.

Tip: Each request results in a distribution and a sync time to the managed devices. For improved performance, limit the number of entries in a file to between 1,000 and 10,000.

Definition

```
repEntries/import
```

Parameters

Parameter	Description	Required
`type`	Address type of the Reputation entry. Possible values include the following: • `ipv4` (default) • `ipv6` • `dns` • `url` Only one type is allowed within a file.	No

Example

The following example uses cURL to import a file with Reputation entries to the SMS. For information on how to format the import file, see *Syntax rules for import files* on page 27.

```
curl -v -k -F "file=@/path/to/file.csv" "http[s]://<sms_server>/repEntries/import
?smsuser=<user_name>&smspass=<password>&type=ipv4"
```

Note: When you request back-to-back imports with files that have 10 or less Reputation entries, the SMS groups those entries to use the `add` method instead to reduce the number of distributions.

Add Reputation entries

Use the `add` method to create a Reputation entry.

Tip: Each request can result in a distribution and a sync time to the managed devices. For improved performance, send requests in bursts up to 1,000 entries in time intervals that allow distributions to complete in a timely manner.

Definition

```
repEntries/add
```

Parameters

Parameter	Description	Required
`ip`	IPv4 or IPv6 address of the Reputation entry.	Only one of the following parameters can be used in the request.
`dns`	DNS address of the Reputation entry.	
`url`	Reputation URL entry.	
`TagData`	One or more tag categories and their values. Must be UTF-8 encoded and separated by a comma (,). **Note:** Reputation entries with a list tag category can include multiple values only when the **Allow Multiple Values?** check box is selected from the **Edit Tag Category** box on the SMS. **Note:** The list values must be separated by ~~~. For example: `MalwareIpType,malwareSource~~~cncHost`	No

Example

The following example uses cURL to add an IPv4 reputation entry with tag categories `MalwareIpType` and `CreatedDate` to the SMS.

```
curl -v -k "http[s]://<sms_server>/repEntries/add
?smsuser=<user_name>&smspass=<password>&ip=1.1.1.1
&TagData=MalwareIpType,infectedHost,CreatedDate,”Jan 22, 2014”"
```

Delete Reputation entries

Use the `delete` method to delete one or more Reputation entries.

Tip: Each request can result in a distribution and a sync time to the managed devices. For optimal performance, delete Reputation entries with a file.

Definition

```
repEntries/delete
```

Parameters

Parameter	Description	Required
`ip`	IPv4 or IPv6 address of the Reputation entry.	Yes with `criteria=entry`
`dns`	DNS address of the Reputation entry.	Yes with `criteria=entry`
`url`	Reputation URL entry.	Yes with `criteria=entry`
`criteria`	Possible values include the following: • `all`: deletes all Reputation entries, including user-defined and RepDV. • `user`: deletes all user-defined entries. • `repdv`: deletes all RepDV entries. • `entry`: deletes specified entries.	Yes

Examples

The following example uses cURL to delete multiple IPv4 and DNS Reputation entries.

```
curl -v -k "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"
```

The following example uses cURL to delete all RepDV entries.

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

Delete reputation entries with a file

When you want to delete a large number of Reputation entries, use `delete` with a file.

Parameters

Parameter	Description	Required
`type`	Address type of the Reputation entry. Possible values include the following: • `ipv4` (default) • `ipv6` • `dns` • `url` Only one type is allowed within a file.	Yes

Example

The following example uses cURL to import a file with Reputation entries to delete on the SMS. For information on how to format the import file, see *Syntax rules for import files* on page 27.

```
curl -v -k -F "file=@/path/to/file.csv" "http[s]://<sms_server>/repEntries/delete
?smsuser=<user_name>&smspass=<password>&type=dns"
```

Query Reputation entries

Use the `query` method to search the Reputation database for one or more user Reputation entries. You can specify up to 10,000 entries in a single request.

The SMS returns all matching entries in the query in UTF-8 encoding. The returned entries are ordered from lowest to highest address, regardless of the order in which they are specified in the query. Each entry is terminated by a newline character.

Definition

```
repEntries/query
```

Parameters

Parameter	Description	Required
`ip`	IPv4 or IPv6 address of the Reputation entry.	Only one of the following parameters can be used in the request.
`dns`	DNS address of the Reputation entry.	
`url`	Reputation URL entry.	

Example

The following example uses cURL to query multiple IPv4 addresses.

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

The preceding query generates the following response:

```
1.1.1.1,AtaHost,myata.device.com,MalwareIpType,infectedHost
1.1.1.2,AtaHost,myata.device.com,ThreatScore,28,MalwareIpType,cncHost~~~infectedHost
```

Virtual segment management

The virtual segment management API enables you to create, update, and delete virtual segments. In addition, you can retrieve a list of virtual segments for SMS-managed devices.

For more information, see the *Security Management System User Guide*.

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 resource in the file must exist on the SMS.

Virtual segment response codes

The following table describes 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

Example responses

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>
```

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

Element	Value	Definition
`description` (optional)	String	Description for the virtual segment
`virtualSegPosition`		Indicates where in the list virtual segment is placed. You define the priority order for a 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
`vlanIdList/vlanList/vlan`		Single element for either a VLAN ID or VLAN range
`vlanIdList/vlanList/vlan/vlanID`	Integer (1 to 4094)	VLAN ID

Element	Value	Definition
`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 referencing a named address group for the destination address

Element	Value	Definition
`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 a 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
`physicalSegments/ physicalSegment/device`		Identifies the device

Element	Value	Definition
`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

Create a virtual segment

Use the `create` method to create a virtual segment with a file.

Definition

```
virtualsegment/create
```

Examples

The following example uses cURL to create a virtual segment.

```
curl -v -k -F "file=@NamedResourceExample.xml"
"http[s]://<sms_server>/virtualsegment/create?smsuser=<user_name>&smspass=<password>"
```

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 a virtual segment* on page 46.

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 a virtual segment

Use the `update` method to update a virtual segment with a file.

Definition

```
virtualsegment/update
```

Parameters

Parameter	Description
`vs`	The name of the virtual segment to be updated on the device and on the SMS.

Example

The following example uses cURL to update a virtual segment named NamedResourceExample.

```
curl -v -k -F "file=@updateAddDeviceSegment.xml"
"http[s]://<sms_server>/virtualsegment/update
?smsuser=<user_name>&smspass=<password>&vs=NamedResourceExample"
```

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

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 a virtual segments

Use the `delete` method to delete a virtual segment.

Parameters

Parameter	Description
`vs`	The name of the virtual segment to be deleted from the device and from the SMS.

Example

The following example uses cURL to delete a virtual segment named namedResourceExample.

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

Retrieve a list of virtual segments

Use the `get` method to retrieve a list of all of the virtual segments on the SMS in XML format. In addition, the request returns the device NAME from the DEVICE table. For more information, see *DEVICE table* on page 61

Note: 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*.

Definition

```
virtualsegment/get
```

Example

The following example uses cURL to get a list of the virtual segments.

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

Remote administration

The remote administration API enables you to backup the SMS database and retrieve SMS software version information.

Backup the SMS database

Use the `backup` resource to create a backup of the SMS database.

Definition

```
smsAdmin/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`

Parameter	Description
`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`)
`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>
```

Retrieve the SMS version

Use the `info` resource to retrieve the SMS software version. The request returns a version number.

Example

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

Vulnerability Scans (eVR)

The Vulnerability Scans (eVR) API enables you import vulnerability scans to the SMS. After you import a vulnerability scan, use the SMS client to:

- View vulnerabilities (listed by CVE) that have been discovered in your network; view which assets are impacted by those vulnerabilities; and view which DV filters can defend those assets from the discovered vulnerabilities.
- Select a profile to quickly highlight DV filters that can protect your assets from the discovered vulnerabilities.
- Flag CVEs for follow-up.
- Track policy changes.
- Adjust profiles to protect your assets.

After you import a vulnerability scan, you can view the scan on the SMS (select **Profiles** > **Vulnerability Scans (eVR)**).

For more information, see the *Security Management System User Guide*.

Vulnerability scan (eVR) specifications

Vulnerability scans must be in a native, comma-separated value (CSV) format before they can be used on the SMS. If you use a supported vulnerability management product, custom converters are available for Qualys®, Rapid7 Nexpose®, and Tenable™ Nessus®.

CSV file specifications

Note the following CSV file specifications (and sequence) for the native SMS-Standard format before you import a vulnerability scan:

- The first line in the CSV file must be the column headers for each of the columns.

- Each row after the header must contain the same number of columns that are in the header.
- Each column must be delimited with a comma.
- The value within each column must be wrapped in double quotes; however, embedded double quotes are not permitted ("This is "invalid" data").
- Each row in a CSV file must be less than 65536 bytes.

Vulnerability scan specifications

The minimum data required for a native SMS-Standard vulnerability scan is:

- **IP Address** - (host IP addresses) The maximum number of host IP address and vulnerability combinations that you can import on the SMS is 10 million. When the SMS reaches the maximum limit, it displays an error message, and you must delete vulnerability scans on the SMS client before you can import a new scan using the eVR API.
- **CVE IDs** - CVE must be in the format CVE-YYYY-NNNN where YYYY is a 4 digit year and NNNN is a sequence number.
- **Severity** - Vulnerabilities are assigned a severity levels to define the urgency associated with remediating each vulnerability. Rankings are based on a variety of industry standards including CVE.

For more information about the native, SMS-Standard fields in the CSV file format on the SMS, select **Profiles** > **Vulnerability Scans (eVR)** > **Import** > **more**.

Import a vulnerability scan (eVR)

Use the `import` method to import a vulnerability scan file that is in native SMS-Standard format.

Definition

```
vulnscanner/import
```

Parameters

Parameter	Description	Required
`vendor`	Name of the vulnerability management vendor. Use the `SMS-Standard` value with the `import` method. For other values, see *Convert a vulnerability scan (eVR)* on page 52.	Yes
`product`	Product name associated with the vulnerability scanner, and can be any value.	Yes

Parameter	Description	Required
`version`	Version of the vulnerability scanning file format, and can be any value.	Yes
`runtime`	Scan start time and end time, and can be a single date or a date range. When entering a date range, you must use a forward slash (/) to separate the scan start and scan end dates. The date format must be yyyy-MM-dd'T'HH:mm:ss.SSS'Z.	Yes

Example

The following example uses cURL to import a vulnerability scan to the SMS in the native SMS-Standard format.

```
curl -v -k -F "file=@vulnScanSampleNativeSMSStandard.csv"
"http[s]://<sms_server>/vulnscanner/import?<smsuser>=<sms_username>
&smspass=<sms_password>&vendor=SMS-Standard&
product=Vulnscanner&version=1.0
&runtime=2014-01-20T13:01:15.255Z/2014-01-20T13:22:14.333Z"
```

Convert a vulnerability scan (eVR)

Use the `convert` method to convert a vulnerability scan file that is not in native SMS-Standard format to import to the SMS.

Definition

```
vulnscanner/convert
```

Parameters

Parameter	Description	Required
`vendor`	Name of the vulnerability management vendor. Possible values include the following: • `Nexpose` • `Qualys-CSV` • `Nessus`	Yes

Parameter	Description	Required
`product`	Product name associated with the vulnerability scanner, and can be any value.	Yes
`version`	Version of the vulnerability scanning file format, and can be any value.	Yes
`runtime`	Scan start time and end time, and can be a single date or a date range. When entering a date range, you must use a forward slash (/) to separate the scan start and scan end dates. The date format must be yyyy-MM-dd'T'HH:mm:ss.SSS'Z.	Yes

Examples

The following example uses cURL to import a vulnerability scan to the SMS in the Nexpose format.

```
curl -v -k -F "file=@vulnScanSampleNexpose.xml"
"http[s]://<sms_server>/vulnscanner/convert?smsuser=<sms_username>
&smspass=<sms_password>&vendor=Nexpose&product=Nexpose&version=1.0
&runtime=2014-01-20T13:01:15.255Z/2014-01-20T13:22:14.333Z"
```

The following example uses cURL to import a vulnerability scan to the SMS in the Qualys-CSV format.

```
curl -v -k -F "file=@vulnScanSampleQualys.csv"
"http[s]://<sms_server>/vulnscanner/convert?smsuser=<sms_username>
&smspass=<sms_password>&vendor=Qualys-CSV&product=Qualys&version=1.0
&runtime=2014-01-20T13:01:15.255Z/2014-01-20T13:22:14.333Z"
```

The following example uses cURL to import a vulnerability scan to the SMS in the Nessus format.

```
curl -v -k -F "file=@vulnScanSampleNessus.nessus"
"http[s]:<sms_server>/vulnscanner/convert?smsuser=<sms_username>
&smspass=<sms_password>&vendor=Nessus&product=Nessus-Sample&version=1.0
&runtime=2014-01-20T13:01:15.255Z/2014-01-20T13:22:14.333Z"
```

Active response

Use the active response API to create and close a response.

By default, no policies can be externally triggered. To enable external triggering, configure the active response policy to allow an SNMP trap or web service to invoke the policy. For more information, see the *Security Management System User Guide*.

Active response best practices

Create a user and policies specifically for this interface to organize which policies are involved with calls that happen externally.

Create a response

Use the `quarantine` method to quarantine an IP address.

Definition

```
quarantine/quarantine
```

Parameters

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

Example

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

Close a response

Use the `unquarantine` method to unquarantine an IP address.

Definition

```
quarantine/unquarantine
```

Parameters

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

Example

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

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 56
- *Events-based packet trace* on page 56

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 61.

```
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.

Set 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.

Database access

Use the SMS web API to access various data, including the SMS data dictionary, table, database schema, status of the web services support, and the version of the SMS web API.

Definition

```
dbAccess/tptDBServlet
```

Parameters

Parameter	Description	Required
`method`	Possible values include the following: • `DataDictionary`: Data dictionary information related to profiles, devices, segments, and virtual segments. • `GetData`: Data from the specified table. • `GetOldestRecord`: The oldest record of the specified table. • `GetNewestRecord`: The newest record of the specified table. • `Schema`: Database schema. • `Status`: Status of the SMS web API support. • `Version`: Version of the SMS web API.	Yes

Usage sequence

Follow this sequence when accessing the SMS database:

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 DVs.
3. Continuously use the `GetData` method, and import the event data into the database.

DataDictionary

Use the `DataDictionary` resource to obtain SMS data dictionary information. For more information about the XML response to a `DataDictionary` request, see *DataDictionary XML response* on page 71.

Definition

```
dbAcess/tptDBServlet?method=DataDictionary
```

Parameters

Parameter	Description	Required
`format`	Possible values include the following: • `sql`(default) • `csv` • `xml`	No
`mode`	Possible values include the following: • `insert`(default) – use with `sql` format. • `update` • `replace`– use with MySQL.	No
`table`	If you do not specify a table, all tables are included. Possible values include the following: • `ACTIONSET` – see *ACTIONSET table* on page 60. • `ALERT_TYPE` – see *ALERT_TYPE table* on page 60. • `DEVICE` – see *DEVICE table* on page 61. • `POLICY` – see *POLICY table* on page 62. • `PRODUCT_CATEGORY` – see *PRODUCT_CATEGORY table* on page 63. • `PROFILE` – see *PROFILE table* on page 63. • `PROFILE_INSTALL_INVENTORY` – see *PROFILE_INSTALL_INVENTORY table* on page 64.	No

Parameter	Description	Required
	• QUARANTINE_NETWORK_DEVICES – see *QUARANTINE_NETWORK_DEVICES table* on page 64. • SEGMENT – see *SEGMENT table* on page 65. • SEGMENT_GROUP – see *SEGMENT_GROUP table* on page 65. • SEVERITY – see *SEVERITY table* on page 66. • SIGNATURE – see *SIGNATURE table* on page 66. • TAXONOMY_MAJOR – see *TAXONOMY_MAJOR table* on page 67. • TAXONOMY_MINOR – see *TAXONOMY_MINOR table* on page 68. • TAXONOMY_PLATFORM – see *TAXONOMY_PLATFORM table* on page 68. • TAXONOMY_PROTOCOL – see *TAXONOMY_PROTOCOL table* on page 68. • THRESHOLD_UNITS – see *THRESHOLD_UNITS table* on page 68. • VIRTUAL_SEGMENT – see *VIRTUAL_SEGMENT table* on page 70.	

Example

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

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

Column	Description
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

Column	Description
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
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

Column	Description
PRODUCT_CATEGORY_ID	Category ID from PRODUCT_CATEGORY table, provided by TippingPoint
PROTOCOL	Well-known PROTOCOL of which this SIGNATURE is part
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_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
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
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

Column	Description
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 64 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

Table name	Response
	DEVICE_GROUP MANAGED See *DEVICE table* on page 61 for more information.
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 65 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 70 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
```

GetData

Use the `GetData` method to request data from specific tables and specify parameters and format.

Definition

```
dbAccess/tptDBServlet?method=GetData
```

Parameters

Parameter	Description	Required
`begin_time`	Type integer. Time is expressed as the number of milliseconds since 01-01-1970 00:00:00 GMT.	Yes
`end_time`	Type integer. Time is expressed as the number of milliseconds since 01-01-1970 00:00:00 GMT.	Yes
`format`	Possible values include the following: • `csv` (default) • `sql` • `xml`	No
`limit`	Type integer. This is the maximum number of values returned. By default, all values are returned.	No
`table`	Possible values include the following: • `ALERTS` • `DDOS_STATS` • `QUARANTINE_HOSTS` • `RATELIMIT_STATS`	Yes

Example

The following example gets data from the ALERTS table with begin and end times in csv format.

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

Events Data

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

- *ALERTS table* on page 76

- *DDOS_STATS table* on page 81
- *QUARANTINE_HOSTS table* on page 86
- *RATELIMIT_STATS table* on page 86

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. 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.

Column	Description
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.
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.

Column	Description
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.
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.

Column	Description

	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 their location in the string.
	Example
	`MESSAGE_PARMS=Austin|Texas`
	`SIGNATURE.MESSAGE=%1 is in %2.`
	The preceding parameters and message generates the following message:
	`Austin is in Texas.`

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.
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).

Column	Description
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.
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.

Column	Description
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

Column	Description
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
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

Column	Description
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
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

Column	Description
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
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

Column	Description
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
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

Column	Description
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

GetNewestRecord

Use the `GetNewestRecord` method to retrieve the newest record of the specified table.

Definition

```
/dbAccess/tptDBServlet?method=GetNewestRecord
```

Parameters

Parameter	Description
`table`	Possible values include the following: • `ALERTS` • `DDOS_STATS` • `QUARANTINE_HOSTS` • `RATELIMIT_STATS`

Example

The following example retrieves the newest record of the ALERTS table.

```
http[s]://<sms_server>/dbAccess/tptDBServlet?method=GetNewestRecord&table=ALERTS
```

GetOldestRecord

Use the `GetOldestRecord` method to retrieve the oldest record of the specified table.

Definition

```
/dbAccess/tptDBServlet?method=GetOldestRecord
```

Parameters

Parameter	Description
table	Possible values include the following: • ALERTS • DDOS_STATS • QUARANTINE_HOSTS • RATELIMIT_STATS

Example

The following example retrieves the oldest record of the ALERTS table.

```
http[s]://<sms_server>/dbAccess/tptDBServlet?
method=GetOldestRecord&table=ALERTS
```

Schema

Use the Schema resource to obtain SMS database schema information.

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

Definition

```
dbAccess/tptDBServlet?method=Schema
```

Parameters

Parameter	Description
database	Only valid for sql format. Possible values include the following: • MySQL (default) • Oracle

Example

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

Status

The `Status` resource returns `OK` if the SMS web API support is enabled and running and `Not Found` if the SMS web API support is not enabled.

Example

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

Version

The `Version` resource returns the version number of the SMS web API.

Example

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

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. 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 76.

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 57
- *Events Data* on page 75

DataDictionary

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

ACTIONSET table on page 60

CATEGORY table on page 60

NOTICE_ACTION table on page 62

POLICY table on page 62

POLICY_GROUP_LOOKUP table on page 63

PROFILE table on page 63

SEVERITY table on page 66

SIGNATURE table on page 66

TAXONOMY_MAJOR table on page 67

TAXONOMY_MINOR table on page 68

TAXONOMY_PLATFORM table on page 68

TAXONOMY_PROTOCOL table on page 68

THRESHOLD_UNITS table on page 68

TPT_DEVICE table on page 69

TPT_SEGMENT table on page 70

TPT_PORT table on page 69

VIRTUAL_SEGMENT table on page 70

Events Data

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

ALERTS table on page 76

DDOS_STATS table on page 81

PORT_TRAFFIC_STATS table on page 85

RATELIMIT_STATS table on page 86

FIREWALL_TRAFFIC_LOG table on page 84

FIREWALL_BLOCK_LOG table **on page 82**

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 94 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

Section	Description	OID
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
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

Section	Description	OID
	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
	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

Section	Description	OID
	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
Temperature	TEMPERATURE	1.3.6.1.4.1.2021.8.1.101.33

TREND MICRO INCORPORATED
225 E. John Carpenter Freeway, Suite 1500
Irving, Texas 75062 U.S.A.
Phone: +1 (817) 569-8900, Toll-free: (888) 762-8736
Email: support@trendmicro.com

www.trendmicro.com
Item Code: APEM57892/170801