# ClearPass Policy Manager 6.3

## Configuration API

The ClearPass API reads and writes a number of configuration elements (each called an Entity) either programmatically or through a script. The API is exposed through an HTTP Post-based mechanism. The API request is in the form of an XML snippet that is posted to a URL hosted by an Admin server on the ClearPass Policy Manager. The response received is an XML snippet.

The request XML and the response XML are structurally defined in an XSD format file. The operations (called Methods) supported are read, write (handles "adds" and "updates"), delete, and name-list based operations:

- Read the names
- Enable status
- Reorder Entity objects

## API Introduction

The ClearPass API reads and writes a number of configuration elements (each called an Entity) either programmatically or through a script. The API is exposed through an HTTP Post-based mechanism. The API request is in the form of an XML snippet that is posted to a URL hosted by an Admin server on the ClearPass Policy Manager. The response received is an XML snippet.

The request XML and the response XML are structurally defined in an XSD format file. The operations (called Methods) supported are read, write (handles "adds" and "updates"), delete, and name-list based operations:

- Read the names
- Enable status
- Reorder Entity objects

## Structure of XML Data

- The root element is *xxx* for a request and *xxx* for a response.
- Sub-element *xxx* will contain information which describes the version of ClearPass (major version followed by the minor version. E.g. 3.0.1) and the time of execution (*exportTime*).
- The next element under root is the body part. The body can either be a list of Entity objects or Filter elements.

## Filter and Criteria Elements

The *Filter* element is used to fetch a list of objects of a specific Entity type. A filter can be used during read and delete operations and can contain a *Criteria* element. A Criteria element must contain the following:

- fieldname – name of the field as present in the XML in which to filter
- filterString – filter string to use during a match of the filter
- match – the operator to be used. For example, the match operator equals/matches the value of the fieldname field in the Entity object using filterString

The following example of a Request XML contains a filter on GuestUser, which contains a Criteria that says to fetch GuestUsers that match the name `kang`.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader version="3.0"/>
```

```
    <Filter entity="GuestUser">
        <Criteria fieldName="name" filterString="kang" match="equals"/>
    </Filter>
</TipsApiRequest>
```

# API Overview

The API is modeled along the lines of a REST-like feature, where each method is represented by a URL. For each operation, the Request XML is posted to a distinct URL identified by the Method. Supported Methods include:

- Read – https://<server>/tipsapi/config/read/<Entity>. The Read Method takes one or more Filter elements and returns a unified list of Entity objects
- Write - https://<server>/tipsapi/config/write/<Entity>. The Write Method takes a list of Entity objects to save. The operation will either add a new object or update an existing one.
- Delete - https://<server>/tipsapi/config/deleteConfirm/<Entity>. The Delete Method consists of a two-step process:
  1. First, the deleteConfirm Method returns a list of identifiers for each of the objects that are to be deleted.
  2. A second request is then made that contains the list of identifiers to delete. The URL for Delete Method is: https://<server>/tipsapi/config/delete/<Entity>

# Authentication

The API Methods require authorization, which is done through BASIC HTTP authentication. The username and password are not passed in the request XML, but they are part of the HTTP call. If the authentication does not go through, an *HTTP Error 401 Unauthorized* message is returned.

The ClearPass Policy Manager Admin credentials should be used for authentication. If the admin does not have the permissions to perform the read, write, delete, etc. operations, then an *HTTP Error 401 Unauthorized* message is returned.

# API Examples

The following examples show how to retrieve, add, update, and remove Guest User values.



## Retrieving a Guest User

### Request

To retrieve a Guest User value post the Request XML to: https://<server>/tipsapi/config/read/GuestUser. Here is a sample XML used to fetch all guest users.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader version="3.0"/>
    <Filter entity="GuestUser"/>
</TipsApiRequest>
```

The following example uses Criteria inside of a Filter.

```
    <Filter entity="GuestUser">
        <Criteria fieldName="name" filterString="kang" match="equals"/>
```

```
    </Filter>
```

### Response

The following example retrieves all guest users that have the name "kang." This Response XML looks similar to the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader exportTime="Thu Sep 30 10:47:26 IST 2010" version="3.0"/>
    <StatusCode>Success</StatusCode>
    <EntityMaxRecordCount>1</EntityMaxRecordCount>
    <GuestUsers>
        <GuestUser enabled="true" expiryTime="2010-12-29 12:24:37.0" startTime="2010-09-29
12:26:08.28" sponsorName="admin" guestType="USER" password="avenda123#" name="kang">
            <GuestUserTags tagName="Company Name" tagValue="Avenda Systems"/>
            <GuestUserTags tagName="Email Address" tagValue="kang@sample.net"/>
            <GuestUserTags tagName="Location" tagValue="Room A"/>
        </GuestUser>
    </GuestUsers>
</TipsApiResponse>
```

## Adding a Guest User Value

### Request

To add a Guest user value, post the Request XML to:
https://<server>/tipsapi/config/write/GuestUser

The Request XML will look similar to the XML received in a read, with the StatusCode, EntityMaxRecordCount, and exportTime omitted:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader version="3.0"/>
    <GuestUsers>
        <GuestUser enabled="true" expiryTime="2010-12-30 12:24:37" startTime="2010-09-30 12
:26:08" sponsorName="admin" guestType="USER" password="avenda123#" name="mike">
            <GuestUserTags tagName="First Name" tagValue="Michael"/>
            <GuestUserTags tagName="Email Address" tagValue="mike@sample.net"/>
            <GuestUserTags tagName="Phone" tagValue="4888888888"/>
        </GuestUser>
    </GuestUsers>
</TipsApiRequest>
```

### Response

The XML response will look similar to the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader exportTime="Thu Sep 30 10:51:27 IST 2010" version="3.0"/>
    <StatusCode>Success</StatusCode>
    <LogMessages>
        <Message>Added 1 guest user(s)</Message>
    </LogMessages>
</TipsApiResponse>
```

## Updating a Guest User Value

The Write Method also handles Update. This is used to determine whether an object passed is already present. Depending on whether the object exists, this method will either add a new object or update the existing object.

### Request

To update a Guest user value, post the Request XML to:
https://<server>/tipsapi/config/write/GuestUser

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader version="3.0"/>
    <GuestUsers>
        <GuestUser enabled="true" expiryTime="2010-12-30 12:24:37" startTime="2010-09-30 12
:26:08" sponsorName="admin" guestType="USER" password="avenda123#" name="mike">
            <GuestUserTags tagName="First Name" tagValue="Michael"/>
            <GuestUserTags tagName="Last Name" tagValue="Penn"/>
            <GuestUserTags tagName="Email Address" tagValue="mike@sample.net"/>
            <GuestUserTags tagName="Phone" tagValue="4888888888"/>
        </GuestUser>
    </GuestUsers>
</TipsApiRequest>
```

### Response for Single Update

The XML response will look similar to the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader exportTime="Thu Sep 30 10:51:27 IST 2010" version="3.0"/>
    <StatusCode>Success</StatusCode>
    <LogMessages>
        <Message>Updated 1 guest user(s)</Message>
    </LogMessages>
</TipsApiResponse>
```

### Response for Multiple Add/Update

In the event that some objects are added and some are updated (for example, if you send five guest user objects), the response XML will look similar to the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader exportTime="Thu Sep 30 10:51:27 IST 2010" version="3.0"/>
    <StatusCode>Success</StatusCode>
    <LogMessages>
        <Message>Added 2 guest user(s)</Message>
        <Message>Updated 3 guest user(s)</Message>
    </LogMessages>
</TipsApiResponse>
```

## Removing a Guest User

The Remove operation is a two-step process that is similar to the Delete process. Use the following to remove a Guest User with the name `kang`.

### Request

Post the Request XML to:
https://<server>/tipsapi/config/deleteConfirm/GuestUser.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader version="3.0"/>
    <Filter entity="GuestUser">
        <Criteria fieldName="name" filterString="kang" match="equals"/>
    </Filter>
</TipsApiRequest>
```

### Response

The XML response will look similar to the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader exportTime="Thu Sep 30 10:47:26 IST 2010" version="3.0"/>
    <StatusCode>Success</StatusCode>
    <EntityMaxRecordCount>1</EntityMaxRecordCount>
    <GuestUsers>
        <GuestUser enabled="true" expiryTime="2010-12-29 12:24:37.0" startTime="2010-09-29
12:26:08.28" sponsorName="admin" guestType="USER" password="avenda123#" name="kang">
            <element-id>GuestUser_kang_MCw</element-id>
            <GuestUserTags tagName="Company Name" tagValue="Avenda Systems"/>
            <GuestUserTags tagName="Email Address" tagValue="kang@avendasys.com"/>
            <GuestUserTags tagName="Location" tagValue="Room A"/>
        </GuestUser>
    </GuestUsers>
</TipsApiResponse>
```

### Request to Extract the Element-IDs

Extract the element-ids, and post the Request XML to:
https://<server>/tipsapi/config/delete/GuestUser

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader version="3.0"/>
    <Delete>
        <Element-Id>GuestUser_kang_MCw</Element-Id>
    </Delete>
</TipsApiRequest>
```

### Response

The response will look similar to the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader exportTime="Thu Sep 30 10:56:00 IST 2010" version="3.0"/>
    <StatusCode>Success</StatusCode>
    <LogMessages>
        <Message>Guest user deleted successfully</Message>
    </LogMessages>
</TipsApiResponse>
```

## Using the Contains Match Operator

The Contains Match operator can be used to fetch multiple items. For example, you can group Guest users that are attending a conference in Sunnyvale (SV) using the format SV_Conf_<user_name>. Then by using a `Criteria` as in the following example, you can fetch the desired group of Guest users:

```
<Filter entity="GuestUser">
    <Criteria fieldName="name" filterString="SV_Conf_" match="contains"/>
</Filter>
```

# Error Handling

In the event of an error or failure during a request, the *StatusCode* is set to Failure. A *TipsApiError* element will be set specifying an *ErrorCode* and a list of Messages.

The following ErrorCodes are defined:

- BadRequest: Method is not supported or is invalid in the URL https://<server>/tipsapi/config/<method>/<Entity>

- InnvalidXml: XML has an invalid structure and contains some extra or missing elements
- IllegalArgument: The Entity type is invalid or does not exist
- InvalidFetchCriteria: A non-existing field name is specified for an entity type, or an invalid filter operation is specified
- ServiceFailure:An internal error occurs in API services
- DependencyBreak: This Entity object is an element in the configuration of some other Entity and is requested for deletion

## Entity Names Supported in Admin API

- Service (services)
- AuthMethod (Authentication Method), AuthSource (Authentication Source)
- Role, LocalUser, GuestUser, StaticHostList, RoleMapping
- PostureInternal (Posture Policies), PostureExternal (Posture Servers), AuditPosture (Audit Servers)
- EnforcementProfile, EnforcementPolicy
- NadClient (Network Devices), NadGroup (Network Device Groups), ProxyTarget
- AdminUser
- SnmpTrapConfig (SNMP Trap Receivers)
- Radius (RADIUS Dictionaries), Posture (Posture Dictionaries)
- SyslogExportData (Syslog Export Filters), ExtSyslog (Syslog Targets)
- AdminReport, PolicySimulation
- ServerConfig – This will return a list of nodes in the ClearPass cluster. This is only supported in read method.

## Other API Methods

ClearPass Policy Manager supports the following additional API methods:


- "Namelist Method" on page 6
- "Reorder Method" on page 6
- "Status Change Method" on page 7


### Namelist Method

URL: https://<server>/tipsapi/config/namelist/<Entity>

The NameList method returns the list of names for all objects created for an Entity type. The request XML contains an *EntityNameList* request passed in the entity type. Multiple EntityNameList requests can be passed for different entity types. In the response, EntityNameList will be populated with the entity names. There is no ordering in the list of names in the response, but for entities that do have an ordering (such as Services), the names are ordered per the list.

### Reorder Method

URL: https://<server>/tipsapi/config/reorder/<Entity>

The Reorder method is available for the Services entity type.

The Reorder method takes a list of object names and Entity types and applies the new order to the list of objects. The request XML contains an *EntityOrderList* that specifies the entity type and the list of Names. The list of Names must contain the names of all elements of the entity type. The new order is returned in the Response XML. Multiple EntityOrderList for differing entity types can be passed in the request.

### Status Change Method

URL: https://<server>/tipsapi/config/status/<Entity>

The Status Change method takes the name-list of disabled and enabled entities of a specific type and changes their status accordingly. The request XML contains an *EntityStatusList* that contains the entity-type and a name-list. Within the name-list, the Enabled elements should first be specified (if any) followed by the Disabled elements. The complete status list is returned in the response.

Policy Manager includes support for multiple EntityStatusList and for different entity-types.

## Advanced Features

Policy Manager includes support for the following advanced features:



### Match Operations

When multiple Filters are specified, the result is a union of the list of elements of all of the filter criteria. For *Match All* criteria, nested Criteria can be specified as *MoreCriteria*. For *Match Any* criteria, multiple Filters with Criteria can be specified for the Entity type. If Criteria is not specified, the operation will fetch all objects of the Entity type.

The following Request fetches all Network Devices that have 192.168.16.* IP address with a vendor specified as IETF.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader version="3.0"/>
    <Filter entity="NadClient">
        <Criteria fieldName="ipAddress" filterString="192.168.16." match="contains">
            <MoreCriteria fieldName="vendorName" filterString="IETF" match="equals">
        </Criteria>
    </Filter>
</TipsApiRequest>
```

The following match operators are supported in Criteria:

- equals – The value of *fieldname* matches the *filterString* exactly.
- notequals – The value of *fieldname* does not match the *filterString* exactly.
- contains – The value of *fieldname* partially matches with the *filterString*, which is case sensitive.
- icontains – This is the case insensitive version of contains.
- belongsto – The value of *fieldname* is one of the values specified in the *filterString*, which can be comma separated in this case.

### Tag/Attribute Search

To enable searches for tagged entities (LocalUser, GuestUser, Endpoint, NadClient and OnboardDevice), add dataType=”ATTRIBUTE” to Criteria/MoreFilterConditions.

If dataType="ATTRIBUTE" is present, then fieldname is the tag name, and fieldString/fieldValue is the tag value.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader version="6.0"/>
    <Filter entity="GuestUser">
        <Criteria fieldName="Device Vendor" filterString="Dell" match="contains" dataType="
ATTRIBUTE">
```

```
            <MoreFilterConditions fieldName="name" fieldValue="test " match="contains"/>
            <MoreFilterConditions fieldName="Device Type" fieldValue ="iPhone" match="conta
ins" dataType="ATTRIBUTE"/>
        </Criteria>
    </Filter>
</TipsApiRequest>
```

## Changing an Entity Name

To change the name of an entity, replace or add the new name in the newName field shown in the following example. This is useful, for example, when a guest requests a new user name.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader version="6.0"/>
    <GuestUsers>
        <GuestUser name="Guest1" newName ="Guest2" approvalStatus="Approved" enabled="true"
expiryTime="2012-10-11 17:45:34 +0545" startTime="2012-10-05 17:45:42 +0545" sponsorName="a
dmin" guestType="USER" password="test"/>
    </GuestUsers>
</TipsApiRequest>
```

## Multiple Sort Options

For additional sort options, a nested feature called “MoreSortOptions” is available. When MoreSortOptions is specified, the result is displayed based on the order of the sort options provided.

Note that Policy Manager support only one Tag (attribute) with the options shown in the following example. Multiple sort options for tags (attributes) are not supported,

```
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
    <TipsHeader version="6.0"/>
    <Filter entity="GuestUser">
        <Criteria fieldName="Location" match="equals" filterString="Bangalore" dataType="AT
TRIBUTE" pageSize="10" pageNumber="1" sortType="asc" sortFieldName="name">
            <MoreSortOptions sortType="asc" sortFieldName="name"/>
            <MoreSortOptions sortType="desc" sortFieldName="expiryTime"/>
        </Criteria>
    </Filter>
</TipsApiRequest>
```