Dell Networking W-ClearPass Policy Manager

Configuration API Guide

Copyright Information

Open Source Code

Certain Aruba products include Open Source software code developed by third parties, including software code subject to the GNU General Public License (GPL), GNU Lesser General Public License (LGPL), or other Open Source Licenses. Includes software from Litech Systems Design. The IF-MAP client library copyright 2011 Infoblox, Inc. All rights reserved. This product includes software developed by Lars Fenneberg, et al. The Open Source code used can be found at this site:

http://www.arubanetworks.com/open_source

Legal Notice

The use of Aruba Networks, Inc. switching platforms and software, by all individuals or corporations, to terminate other vendors' VPN client devices constitutes complete acceptance of liability by that individual or corporation for this action and indemnifies, in full, Aruba Networks, Inc. from any and all legal actions that might be taken against it with respect to infringement of copyright on behalf of those vendors.

Contents

The Dell Networking W-ClearPass Policy Manager Configuration Application Programming Interface (API) is used to read and write a number of configuration elements (known as **Entities**) either programmatically or using a script. ClearPass configuration API allows you to configure or modify the entities in ClearPass without logging into the admin User Interface (UI). For example, when you create a new user in database, you may want to create a guest user automatically. You can use ClearPass Policy Manager configuration API to automate this task. The API is available through an HTTP POST based mechanism. Both the API request and response are in the form of an XML snippet that is posted to a URL hosted by an Admin server on ClearPass. The XML request and the XML response are structurally defined in an XSD format file.

The **Read**, **Write** (handles additions and updates), and **Delete** operations (known as **Methods**) are supported in ClearPass Policy Manager Configuration API. You can use these methods to perform the following name-list based operations:

- **NameList** – This method returns the list of names for all objects created for an entity type.
- **Reorder** – This method receives a list of names of objects of the entity type and applies the new order to the list of objects.
- **Status Change** - This method gets the name-list of disabled and enabled entities of a specific type and changes the status of the entities appropriately.

Every XML request must conform to the Dell Networking W-ClearPass Policy Manager Configuration API XML schema.

Structure of XML Data

The following elements define the structure of XML data:

- **Root**: The root element is **<TipsApiRequest>** for a request and **<TipsApiResponse>** for a response.
- **Sub-element**: The sub-element element **<TipsHeader>** describes the version of ClearPass Policy Manager (major version followed by the minor version, for example, 3.0.1). The **Sub-element** is the container object that can be controlled by adding and modifying attributes. The sub-element in the XML request contains only the Version number and the sub-element in the XML response contains the Version number, Time of execution (exportTime), and Entity types.
- **Body**: The body element describes the child elements of XML data. The body contains the **Filter** elements in the XML request and a list of **Entity** objects in the XML response.

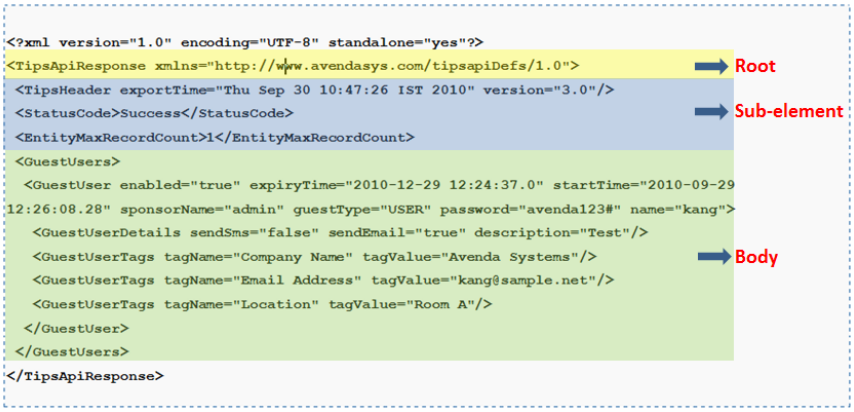
The following figure describes the structure of XML data in an XML request:

Figure 1: *Structure of XML Data - XML Request*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">          → Root
<TipsHeader version="3.0"/>                                                → Sub-element
<Filter entity="NadClient">
<Criteria fieldName="ipAddress" filterString="192.168.16.x" match="contains">
<MoreFilterConditions fieldName="name" fieldValue="IETF" match="equals"/>  → Body
</Criteria>
</Filter>
</TipsApiRequest>
```

The following figure describes the structure of XML data in an XML response:

Figure 2: *Structure of XML Data - XML Response*



Filter Elements

Use the **Filter** element to fetch a list of objects of a specific entity. You can use a filter to perform the **Read** and **Delete** operations. A filter contains a **Criteria** element that includes the following:

- **fieldname** – Specifies the name of the field present in XML that needs to be filtered.
- **filterString** – Specifies the string that is used to match the filter during a match of the filter.
- **match** – Specifies 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 is the example of an XML request contains a filter on Guest user with a **Criteria** that initiates requests to fetch Guest users 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" source="Guest"/>
<Filter entity="GuestUser">
<Criteria fieldName="name" filterString="kang" match="equals"/>
</Filter>
</TipsApiRequest>
```

The Dell Networking W-ClearPass Policy Manager Configuration API is modeled similar to a Representational State Transfer (REST) API, where each method is represented by a URL. For each operation, XML request is posted to a different URL identified by the following methods:

- **Read** –The **Read** method gets one or more filter elements and returns a unified list of Entity objects. The URL for the **Read** method is https://<server>/tipsapi/config/read/<Entity>.
- **Write** - The **Write** method gets a list of Entity objects to save. The operation either adds a new object or updates an existing one. The URL for the **Write** method is https://<server>/tipsapi/config/write/<Entity>.
- **Delete** - The **Delete** method functions in the following two steps:
  - Initially, the **deleteConfirm** method returns a list of identifiers for each object that needs to be deleted. The URL for the **deleteConfirm** method is https://<server>/tipsapi/config/deleteConfirm/<Entity>.
  - Creates a second request that contains the list of identifiers to delete. The URL for the **Delete** method is https://<server>/tipsapi/config/delete/<Entity>.

Entity Names Supported in ClearPass Configuration API

The following table describes the Entity names supported in the Configuration API:

Table 1: *Supported Entity Names in Configuration API*

API Entity Name Settings	Description
Service	Specifies a Service and associated entities.
AuthMethod	Specifies the Authentication Method to authenticate the user or device against an Authentication Source.
AuthSource	Specifies the identity store (Active Directory, LDAP Directory, SQL DB, and Token Server) against which users and devices are authenticated.
LocalUser	Specifies the Local User Repository.
Endpoint	Specifies the Endpoint device details. **NOTE:** Profile information is not supported in API.
StaticHostList	Comprises of a list of MAC and IP addresses. These can be used as white-lists or blacklists to control access to the network.
Role	Specifies a set of roles assigned by the role mapping policy.
RoleMapping	Specifies the Role-Mapping Policy.
PostureInternal	Specifies the Internal Posture Policy that tests requests against Internal Posture rules to assess health.
PostureExternal	Specifies the **External Posture** server.

Table 1: *Supported Entity Names in Configuration API (Continued)*

API Entity Name Settings	Description
AuditPosture	Specifies the **Audit Posture** servers such as NMAP and Nessus.
EnforcementPolicy	Specifies the Enforcement Policy that applies conditions (roles, health, and time attributes) against specific values associated with those attributes to determine the Enforcement Profile.
EnforcementProfile	Specifies the Enforcement Profiles contain attributes that define a client's scope of access for the session.
NadClient	Specifies the Network Device.
NadGroup	Specifies the Network Device Group.
ProxyTarget	Specifies the RADIUS request that needs to be proxied to another RADIUS server.
Simulation	Specifies the Policy simulations that allow policies to be verified before they are deployed.
AdminUser	Specifies the Admin User Repository.
AdminPrivileges	Specifies the Admin User Privileges.
ServerConfig	Provides the Server Configuration details. **NOTE:** Only the **Read** method is permitted.
SnmpTrapConfig	Specifies SNMP Trap Receivers.
ExtSyslog	Specifies the session data, audit records, and event records that can be sent to one or more syslog targets (servers).
DataFilter	Specifies the Data Filters used to filter records in Access Tracker and Syslog messages.
SyslogExportData	Specifies the Syslog Export Filters to notify Policy Manager where to send this information and what type of information should be sent through Data Filters.
ContextServer	Specifies the Endpoint Context Server.
ContextServerAction	Specifies the Endpoint Context Server Actions dictionary to configure actions that are performed on endpoints.
RADIUS Dictionary	Specifies the RADIUS vendor attributes dictionary.
PostureDictionary	Specifies the posture attributes dictionary.
TacacsServiceDictionary	Specifies the TACACS+ Service attributes dictionary.
TagDictionary	Specifies the Entity Tag Attributes dictionary.

Table 1: *Supported Entity Names in Configuration API (Continued)*

API Entity Name Settings	Description
TagDefinition	Specifies the Entity Tag Definitions.
GuestUser	Specifies the Guest accounts managed by ClearPass Guest module.
OnboardDevice	Specifies the Onboard devices managed by ClearPass Onboard module.

The source attribute with the value **Guest** must be used for the **GuestUser** and **OnboardDevice** entity types. For other entity types, do not need to include the source attribute.

Authentication

The API Methods require authorization, which is performed using HTTP Basic authentication. The username and password are not passed in the XML request, however, they are part of the HTTP call. If the authentication is unsuccessful, the **401 Unauthorized** http error message appears. The Dell Networking W-ClearPass Policy Manager Administrator credentials must be used for authentication. If the administrator does not have the permissions to perform the read, write, and delete operations, the **401 Unauthorized** http error message appears.

This section provides the examples of XML request and response used to perform the following tasks:

- Retrieving a Guest User Value
- Retrieving a Local User Value
- Adding a Guest User Value
- Updating a Guest User Value
- Removing a Guest User
- Using the Contains Match Operator

Retrieving a Guest User Value

Post the XML request to the URL https://<server>/tipsapi/config/read/GuestUser. The following is an example of the XML request 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" source="Guest"/>
<Filter entity="GuestUser"/>
</TipsApiRequest>
```

The source attribute with the value **Guest** must be used for the **GuestUser** and **OnboardDevice** entity types. For other entity types, do not need to include the source attribute.

Retrieving a Local User Value

Post the XML request to https://<server>/tipsapi/config/read/LocalUser. The following is an example of XML request used to fetch all local 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="LocalUser"/>
</TipsApiRequest>
```

The source attribute with the value **Guest** must be used for **GuestUser** and **OnboardDevice** entity types. For other entity types, do not include the source attribute.

The following is an example with **Criteria** in a filter:

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

The following is an example of the XML response that retrieves all Guest users with the name **kang**:

```
<?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">
<GuestUserDetails sendSms="false" sendEmail="true" description="Test"/>
<GuestUserTags tagName="Company Name" tagValue="Avenda Systems"/>
<GuestUserTags tagName="Email Address" tagValue="kang@sample.net"/>
<GuestUserTags tagName="Location" tagValue="Room A"/>
</GuestUser>
</GuestUsers>
</TipsApiResponse>
```

For other entity types, do not need to include the source attribute. If **Guest** description is present in the XML request, the **GuestUserDetails** element is displayed in the **Guest** details.

Adding a Guest User Value

Post the XML request to the URL https://<server>/tipsapi/config/write/<GuestUser>. The following is an example of the XML request that is similar to the XML response received in the **Read** method 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" source="Guest"/>
<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">
<GuestUserDetails sendSms="false" sendEmail="false" description="Test"/>
<GuestUserTags tagName="First Name" tagValue="Michael"/>
<GuestUserTags tagName="Email Address" tagValue="mike@sample.net"/>
<GuestUserTags tagName="Phone" tagValue="4888888888"/>
</GuestUser>
</GuestUsers>
</TipsApiRequest>
```

For Guest description, you must include the **GuestUserDetails** element as described in the example above. You can set the **sendSms** and **sendEmail** attribute values to **false** as these values are not used by ClearPass Guest.

The following is an example of the XML response:

```
<?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 handles the **Update** operation and determines whether a passed object in the XML request is already present or not. Depending on presence of the passed object, a new object is added or the existing object is updated. Post the request XML request to the URL https://<server>/tipsapi/config/write/<GuestUser>. The following is an example of the XML request:

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

The following is an example of the XML response:

```
<?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>
```

The following is an example of the XML response with some objects added and updated:

```
<?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 involves two steps similar to the **Delete** operation. Use the following steps to remove a Guest user with the name **kang**:

1. Post the XML request to the URL https://<server>/tipsapi/config/deleteConfirm/<GuestUser> as described in the following example:

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

The following is an example of the XML response:

```
<?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>
```

2. Extract the element-Ids and post the XML request to the URL https://<server>/tipsapi/config/delete/<GuestUser> as described in the following example:

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

The following is an example of the XML response:

```
<?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

Use the **Contains** match operator to fetch more than one item. For example, you can group Guest users who attend a conference in Sunnyvale (SV) using the format **SV_Conf_<user_name>**. You can fetch the required group of Guest users using the **Criteria** as described in the following example:

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

With the examples provided in this section, you can retrieve, add, update, and remove the Guest User value and the Local User value.

Error Handling

When there is an error or failure during a request, the **StatusCode** is set to **Failure**. A **TipsApiError** element is set with an **ErrorCode** and a list of messages. The following error codes are defined in Admin API:

- **BadRequest**: This error occurs when the method described in the URL https://<server>/tipsapi/config/<method>/<Entity> is not supported or is invalid.
- **InvalidXml**: This error occurs when XML has an invalid structure and contains some additional or missing elements.
- **IllegalArgument**: This error occurs when the Entity type is invalid or does not exist.
- **InvalidFetchCriteria**: This error occurs when a field name specified does not exist for an entity type or the specified filter operation is invalid.
- **ServiceFailure**: This error occurs in case of an internal error occurs in API services.
- **DependencyBreak**: This error occurs when the Entity object is an element of some other Entity and is requested for deletion.

The following is an example of the error message occurs when a field name specified does not exist for an entity type or the specified filter operation is invalid:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
<TipsHeader exportTime="Wed May 28 15:31:41 IST 2014" version="6.3"/>
<StatusCode>Failure</StatusCode>
<TipsApiError>
<ErrorCode>InvalidFetchCriteria</ErrorCode>
<Message>Invalid FieldName. 'macaddress' is not a field of Endpoint entity</Message>
</TipsApiError>
</TipsApiResponse>
```

The source attribute with the value **Guest** must be used for the **GuestUser** and **OnboardDevice** entity types. For other entity types, do not need to include the source attribute.

The following other API methods are available in the ClearPass Configuration API:

- NameList
- Reorder
- Status Change

NameList

The **NameList** method returns the list of names for all objects created for an Entity type. The XML request contains an **EntityNameList** request passed in the entity-type. Multiple **EntityNameList** requests can be passed for different Entity types. In the XML response, **EntityNameList** is populated with the entity-names. The list of names in the XML response is not displayed in a specific order. However, for the entities that have a specific order (for example, **Services**), the names are populated in the order as specified in the **EntityNameList**.

The URL for the **NameList** method is https://<server>/tipsapi/config/namelist/<Entity>. The following is an example of the XML request for the **NameList** method:

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

The following is an example of the XML response for the **NameList** method:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0"><TipsHeader
exportTime="Wed May 28 15:39:01 IST 2014" version="6.3"/>
<StatusCode>Success</StatusCode>
<EntityNameList entity="Service"><Name>[Policy Manager Admin Network Login Service]
</Name><Name>[AirGroup Authorization Service]</Name><Name>[Aruba Device Access Service]
</Name><Name>[Guest Operator Logins]</Name><Name>test 802.1X Wireless</Name>
</EntityNameList>
</TipsApiResponse>
```

Reorder

The **Reorder** method receives a list of names of objects of the entity type and applies the new order to the list of objects. The XML request contains an **EntityOrderList** that should specify the entity-type and a list of names. This list should contain the names of all elements of the entity-type. The new order is returned in the XML response. Multiple **EntityOrderList** for different entity-types can be passed in the request. The **Reorder** method is available for the **Services** entity-type.

The URL for the **Reorder** method is https://<server>/tipsapi/config/reorder/<Entity>. The following is an example of the XML request for the **Reorder** method:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
<TipsHeader version="6.3"/>
<EntityOrderList entity="Service"><Name>[Aruba Device Access Service]</Name>
<Name>[Guest Operator Logins]</Name><Name>test 802.1X Wireless</Name>
<Name>[Policy Manager Admin Network Login Service]</Name>
<Name>[AirGroup Authorization Service]</Name></EntityOrderList>
```

```
</TipsApiRequest>
```

The following is an example of the XML response for the **Reorder** method:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
<TipsHeader exportTime="Wed May 28 15:45:24 IST 2014" version="6.3"/>
<StatusCode>Success</StatusCode>
<LogMessages><Message>Services have been reordered successfully</Message></LogMessages>
<EntityOrderList entity="Service"><Name>[Aruba Device Access Service]</Name>
<Name>[Guest Operator Logins]</Name><Name>test 802.1X Wireless</Name>
<Name>[Policy Manager Admin Network Login Service]</Name>
<Name>[AirGroup Authorization Service]</Name>
</EntityOrderList>
</TipsApiResponse>
```

Status Change

The **Status Change** method gets the name-list of disabled and enabled entities of a specific type and changes the status of the entities appropriately. The XML request contains an **EntityStatusList** that includes the entity-type and a name-list. You must specify the **Enabled** elements first and then the **Disabled** elements within the name-list. The status list of the entity is returned in the XML response.

Multiple EntityStatusList for different entity types are supported.

The URL for the **Status Change** method is https://<server>/tipsapi/config/status/<Entity>. The following is an example of the XML request for the **Status Change** method:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiRequest xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
<TipsHeader version="6.3"/>
<EntityStatusList entity="Service">
<Enabled>[Aruba Device Access Service]</Enabled>
<Enabled>[Guest Operator Logins]</Enabled>
<Disabled>test 802.1X Wireless</Disabled>
<Disabled>[Policy Manager Admin Network Login Service]</Disabled>
</EntityStatusList>
</TipsApiRequest>
```

The following is an example of the XML response for the **Status Change** method:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
<TipsHeader exportTime="Wed May 28 16:08:13 IST 2014" version="6.3"/>
<StatusCode>Success</StatusCode>
<LogMessages><Message>Status successfully changed</Message></LogMessages>
<EntityStatusList entity="Service">
<Enabled>[AirGroup Authorization Service]</Enabled>
<Enabled>[Aruba Device Access Service]</Enabled>
<Enabled>[Guest Operator Logins]</Enabled>
<Disabled>[Policy Manager Admin Network Login Service]</Disabled>
<Disabled>test 802.1X Wireless</Disabled>
</EntityStatusList>
</TipsApiResponse>
```

Advanced Match Operations

When multiple filters are specified, the result can be a combination of the list of elements of all of the filter criteria. For **Match All** criteria, specify the nested criteria as **MoreFilterConditions**. For **Match any** criteria, multiple Filters with criteria can be specified for the entity type. If a criteria is not specified, then the **Advanced Match** operation fetches all objects of the entity type. The following example describes the XML request that fetches all network devices with the IP address 192.168.16.x and vendor 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.x" match="contains">
<MoreFilterConditions fieldName="name" fieldValue="IETF" match="equals"/>
</Criteria>
</Filter>
</TipsApiRequest>
```

The following entity types support tag attributes:

- Endpoint
- Device
- LocalUser
- GuestUser

To filter based on the tag attributes, include an additional attribute called dataType=”ATTRIBUTE” for that filter condition as described in the following example:

```
<?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">
<MoreFilterConditions fieldName="TagName" fieldValue="TagValue" match="equals"
dataType="ATTRIBUTE"/>
</Criteria>
</Filter>
</TipsApiRequest>
```

The following match operators are supported in a 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**: 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.

With the XML request and response examples given in this section, you can use the **Advanced Match** operation to fetch all objects of an Entity type.

This chapter describes the best practices to be followed to use the Dell Networking W-ClearPass Policy Manager Configuration API. Presently, the support for paged results for entities is not available. This can impact the system when the API query for entities with more than 50 entries. For example, a bulk query to get all Endpoints or ClearPass Guest accounts when there are hundreds of entries present in the system is challenging. To manage ClearPass Guest accounts, Onboard certificates, ClearPass Guest, and Onboard application features, it is recommended to use the Guest APIs supported by ClearPass Guest.

Bulk Access for Endpoints and Guest Accounts

For better query performance and minimal load on the system, you must use the bulk query cautiously.

Entities such as Endpoints and ClearPass Guest users can grow in thousands depend on the deployment. These entities support tag attributes which are custom key-value pairs added by the system or Administrator that provides more context to the entity. A bulk query to fetch all the details of the endpoints or ClearPass Guest users in the system can impact on the system performance. Alternatively, you can primarily use the **NameList** query followed by the query on individual details for each name present in the NameList response depends on the specific endpoint.

1. Use the following command to fetch the list of MAC addresses for the endpoints present in the system:

```
wget --no-check-certificate --http-user=<USER> --http-password=<PASSWORD> --post-file=in.xml https://CPPM-Server/tipsapi/config/namelist/Endpoint
```

The following is an example of the XML request for the **Namelist** method:

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

The following is an example of the XML response for the **Namelist** method:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
<TipsHeader exportTime="Mon Jul 21 13:37:13 IST 2014" version="6.4"/>
<StatusCode>Success</StatusCode>
<EntityNameList entity="Endpoint">
<Name>000c29eff62f</Name>
<Name>001122aabbcc</Name>
</EntityNameList>
</TipsApiResponse>
```

2. Use the following command to fetch the list of endpoints for a specific MAC address:

```
wget --no-check-certificate --http-user=<USER> --http-password=<PASSWORD> https://CPPM-Server/tipsapi/config/read/Endpoint/equals?macAddress=000c29eff62f
```

The following is an example of the XML response for the **Namelist** method:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsApiResponse xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
<TipsHeader exportTime="Mon Jul 21 14:50:09 IST 2014" version="6.4"/>
<StatusCode>Success</StatusCode>
<EntityMaxRecordCount>1</EntityMaxRecordCount>
<Endpoints>
```

```
<Endpoint macAddress="000c29eff62f" status="Known"/>
<EndpointTags tagValue="true" tagName="Encryption Enabled"/>
<EndpointTags tagValue="PDA 2" tagName="Phone Number"/>
<EndpointTags tagValue="MobileIron" tagName="Source"/>
<EndpointTags tagValue="3fbe0a80-e7d2-4048-bd2e-62aec232a236"     tagName="MDM Identifie
r"/>
<EndpointTags tagValue="Bala" tagName="Display Name"/>
<EndpointTags tagValue="iPad 2" tagName="Model"/>
<EndpointTags tagValue="true" tagName="MDM Enabled"/>
<EndpointTags tagValue="balu" tagName="Owner"/>
<EndpointTags tagValue="Installed" tagName="Required App"/>
<EndpointTags tagValue="b786da8ca3969e0134f058ca5efe94687ab7f31f"   tagName="UDID"/>
<EndpointTags tagValue="iOS 7.0" tagName="OS Version"/>
<EndpointTags tagValue="PDA" tagName="Carrier"/>
<EndpointTags tagValue="false" tagName="Compromised"/>
<EndpointTags tagValue="Corporate" tagName="Ownership"/>
<EndpointTags tagValue="false" tagName="Blacklisted App"/>
<EndpointTags tagValue="Apple" tagName="Manufacturer"/>
</Endpoint>
</Endpoints>
</TipsApiResponse>
```

Usage of Advanced Match Operations

The complex query supported in the **Advanced Match Operations** must be used with care.

The number of entities and the associated tag attributes with each entity can impact the performance. You can use the API to query based on tag attributes when the queries are not repeated.

Admin Accounts for API Access

The Admin users configured in the ClearPass Policy Manager can only use the API access. It is recommended to create a separate user for API access than using the default admin user account. To create a new user for API access, update the password of the default apiadmin user account or create a new Admin user with only API access privileges. This ensures that all API actions are tracked through the **Audit Viewer** page for this user account. Additionally, restrictions to specific Entities can be enforced by defining a custom admin privilege level and creating API admin users with that privilege level. This ensures that the API account included in client scripts secure the confidential information in the ClearPass Policy Manager system.