Using the vManage REST APIs

The Viptela software provides a REST API, which is a programmatic interface for controlling, configuring, and monitoring the Viptela devices in an overlay network. You access the REST API through the vManage web server.

A REST API is a web service API that adheres to the REST, or Representational State Transfer, architecture. The REST architecture uses a stateless, client–server, cacheable communications protocol. The Viptela vManage NMS web server uses HTTP and its secure counterpart, HTTPS, as the communications protocol. REST applications communicate over HTTP or HTTPS using standard HTTP methods to make calls between network devices. These standard HTTP methods include:

- GET—Retrieve or read information.
- PUT—Update an object.
- POST—Create an object.
- DELETE—Remove an object.

REST is a simpler alternative to mechanisms such as remote procedure calls (RPCs) and web services such as Simple Object Access Protocol (SOAP) and Web Service Definition Language (WSDL). The REST architecture has not been formally defined by any standards bodies.

For more information about REST, see this Wikipedia page.

This article explains basic REST terminology, describes how to use the Viptela REST API in conjunction with the vManage NMS, and provides some examples of script workflows for performing overlay-related network operations.

Available API Calls

The Viptela REST API calls expose the functionality of Viptela software and hardware features and of the normal operations you perform to maintain Viptela devices and the overlay network itself. In REST API terminology, each of these features or operations is called a resource. A resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it.

Resources are grouped into collections. Each collection contains a single type of resource, and so is homogeneous. In the Viptela REST API, the collection of resources is present at the top level of the API. The Viptela REST API resources are grouped into the following collections:

<table>
<thead>
<tr>
<th>Resource Collection</th>
<th>Resources for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration</td>
<td>Managing users and user groups, viewing audit logs, and managing the local vManage server.</td>
</tr>
<tr>
<td>Certificate Management</td>
<td>Managing certificates and security keys.</td>
</tr>
<tr>
<td>Configuration</td>
<td>Creating feature and device configuration templates, retrieving the configurations in existing templates, and</td>
</tr>
</tbody>
</table>

https://sdwan-docs.cisco.com/Product_Documentation/Command_Reference/Command_Reference/vManage_REST_APIs/v…
Created on: Mon, 21 Oct 2019 04:15:39 GMT
Generated by: Anonymous
### Resource Data

The Viptela REST API uses the JavaScript Object Notation (JSON) data model to represent the data associated with a resource. JSON has three types of data:

- **Scalar**—Data types with a single value. Scalar types include numbers, strings, and booleans.
- **Array**—Ordered lists of values of an arbitrary type.
- **Object**—Unordered set of key:value pairs (also referred to as attributes), where key is a string and the value can have an arbitrary type.

For more information about JSON, see the [JSON Data Interchange Standard](https://www.json.org).

### REST Methods

In a REST API, methods are executed on resources and collections. The Viptela REST API uses the following standard REST methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Scope</th>
<th>Shown As</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>Resource</td>
<td>Red</td>
<td>Delete a resource.</td>
</tr>
<tr>
<td>GET</td>
<td>Collection, Resource</td>
<td>Blue</td>
<td>Retrieve (or read) all resources in a collection or retrieve a single resource.</td>
</tr>
<tr>
<td>POST</td>
<td>Collection</td>
<td>Green</td>
<td>Create a new resource in a collection.</td>
</tr>
<tr>
<td>PUT</td>
<td>Resource</td>
<td>Brown</td>
<td>Update a resource.</td>
</tr>
</tbody>
</table>
These four resources are sometimes referred to as CRUD operations, for create, read, update, and delete.

**Viptela API Library and Documentation**

A common REST principle is that APIs should be self-descriptive and self-documenting. The resource collections and resources in the Viptela REST API are self-documenting, describing the data you need to make a call and the response from each call. However, the collections and resources do assume that you are familiar with the Viptela overlay network concepts, software and hardware features, and capability.

The vManage REST API library and documentation are bundled with and installed on the vManage web application software. To access the API documentation from a web browser, use this URL:

```
https://ip-address:port/apidocs
```

`ip-address` is the IP address of the vManage server, and `port` is the port used for the vManage server, which is typically 8443. For example, if the vManage web server has the URL https://192.168.48.83:8443, you access the vManage REST API at https://192.168.48.83:8443/apidocs.

**Performing REST API Operations on a vManage Web Server**

To transfer data from a vManage web server using a utility such as Python, follow this procedure:

1. Establish a session to the vManage web server.
2. Issue the desired API call.

See the following sections for more information.

**Establish a Session to the vManage Server**

When you use a program or script to transfer data from a vManage web server or perform operations on the server, you must first establish an HTTPS session to the server. To do this, you send a call to log in to the server with the following parameters:

- **URL to send the request to**—Use https://{vmanage-ip-address}/j_security_check, which performs the login operation and security check on the vManage web server at the specified IP address.

- **Request method**—Specify a Post request.

- **API call input**—The input is an application, so for the Content Type, specify application/x-www-form-urlencoded.

- **API call payload**—The payload contains the username and password in the format `j_username=username&j_password=password`.

The following example program shows how to use Python to establish a session to a given vManage server, with the username and password indicated on the command line. It runs a /device call, then prints the response, a JSON object. This code uses the Python requests library from [www.python-requests.org](http://www.python-requests.org). The Viptela API has been tested with

https://sdwan-docs.cisco.com/Product_Documentation/Command_Reference/Command_Reference/vManage_REST_APIs/v…
Python programming language version 2.7.9 and Python request library version 2.9.1.

```python
class rest_api_lib:
    def __init__(self, vmanage_ip, username, password):
        self.vmanage_ip = vmanage_ip
        self.session = {}
        self.login(self.vmanage_ip, username, password)

    def login(self, vmanage_ip, username, password):
        base_url_str = 'https://%s/'%vmanage_ip
        login_action = '/j_security_check
        login_data = {'j_username': username, 'j_password': password}
        login_url = base_url_str + login_action
        sess = requests.session()
        login_response = sess.post(url=login_url, data=login_data, verify=False)
        if '<html>' in login_response.content:
            print "Login Failed"
            sys.exit(0)
        self.session[vmanage_ip] = sess

    def get_request(self, mount_point):
        url = "https://%s:8443/dataservice/%s"%(self.vmanage_ip, mount_point)
        print url
        response = self.session[self.vmanage_ip].get(url, verify=False)
        data = response.content
        return data

    def post_request(self, mount_point, payload, headers={'Content-Type': 'application/json'}):
        url = "https://%s:8443/dataservice/%s"%(self.vmanage_ip, mount_point)
        payload = json.dumps(payload)
        response = self.session[self.vmanage_ip].post(url=url, data=payload, headers=headers, verify=False)
        data = response.content
        return data

def main(args):
    if not len(args) == 3:
        print __doc__
        return
    vmanage_ip, username, password = args[0], args[1], args[2]
    obj = rest_api_lib(vmanage_ip, username, password)
    # Example request to get devices from the vmanage "url=https://vmanage.viptela.com/dataservice/device"
    response = obj.get_request('device')
    print response
    # Example request to make a Post call to the vmanage "url=https://vmanage.viptela.com/dataservice/device/action/rediscover"
    payload = {'action': 'rediscover', 'devices': [{'deviceIP': '172.16.248.105'}, {'deviceIP': '172.16.248.106'}]}
    response = obj.post_request('device/action/rediscover', payload)
    print response

if __name__ == '__main__':
    sys.exit(main(sys.argv[1:]))
```
API Calls and Responses

To issue a REST API call, specify the URL of the call in the browser’s URL field. Specify the URL in the following format:

```
https://vmanage-ip-address/api-call-url
```

If you are logged in to a vManage web server, you can place this call directly in the browser’s URL field.

To illustrate how to make API calls and the API responses, we use the call that retrieves a list of all devices in the network. To retrieve this list, use the following URL:

```
https://vmanage-ip-address/dataservice/device
```

You can find the documentation for this call under the Monitoring – Device Details resource collection. This call is a GET request, and it also indicates the URL to use to send the request.

To issue this call from a browser, enter the URL `https://vmanage-ip-address/dataservice/device`. The call returns a JSON object that is large because it contains device information for all devices in the network. The output is returned on a single line.

To filter the results of this call so you get information only for a single device, you add query string parameters. For example, to limit the previous GET response to a single device, specify a system IP address. If you change the URL to `https://vmanage.corp.viptela.com/dataservice/device?system-ip=172.16.248.198`, the JSON object returned is for only that device, so the output is much shorter.

Additional Information

Overview of Bulk API Operations
vManage How-Tos