VXOA REST API Guide – Common Tasks

Version 7.2, April 2015

silver peak™

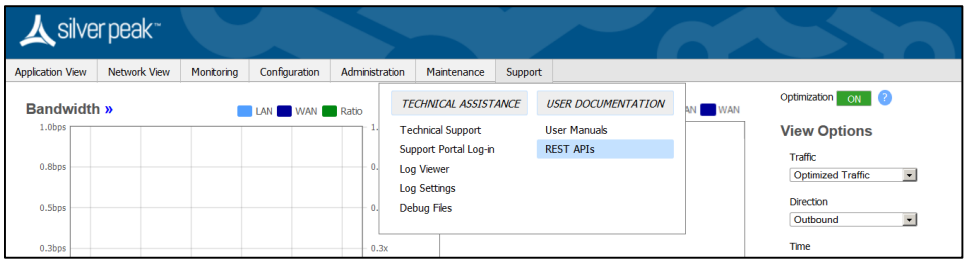
Contents

Scope of This Guide

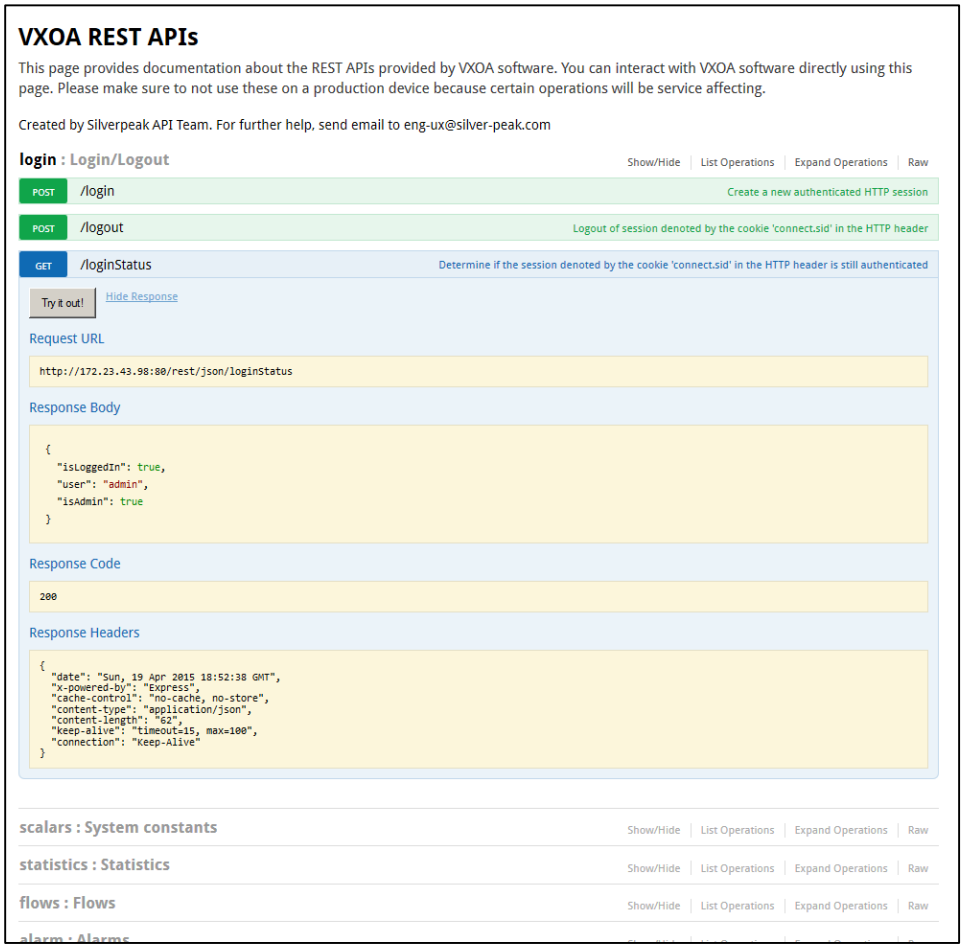
This guide documents REST APIs provided by VXOA software. Note that this guide only provides documentation for a subset of REST APIs representing the most common initial configuration tasks.

Where to get more info

For a comprehensive and interactive guide to VXOA REST APIs, log onto a Silver Peak appliance running VXOA 7.2 or later, and navigate to the embedded "REST APIs" user documentation found within the "Support" tab:



The embedded documentation provides a fully interactive help experience:

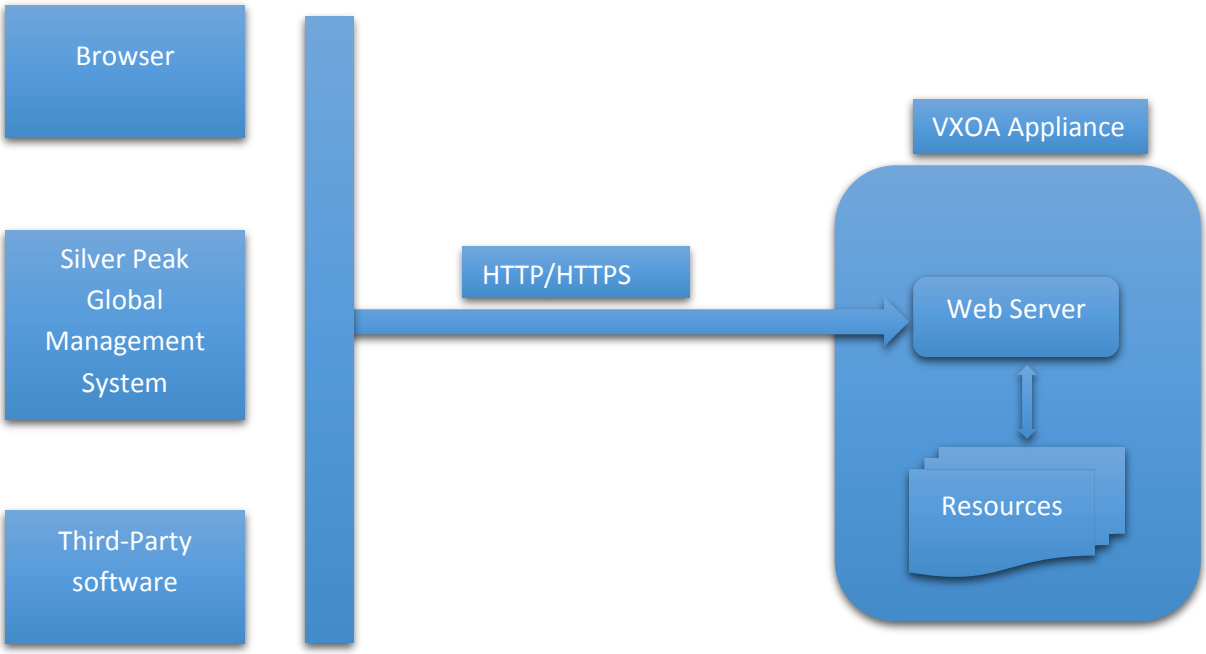


API Overview

VXOA appliances feature an API that supports configuration and monitoring functions using HTTP and HTTPS as transport.

Architecture

All resources are secured using the authentication scheme configured on the VXOA appliance (this includes RADIUS and TACACS support). Access the API with the same credentials that you use for VXOA appliance login.



The above diagram illustrates the communication between the REST API client and the VXOA appliance. The following sections describes how to configure and monitor a VXOA appliance using this protocol.

Data formatting

This API follows REST guidelines. Various VXOA functions are exposed as resources which can be manipulated as follows:

1. Resources can be received using HTTP GET operation
2. Resources can be modified using HTTP POST, PUT, and DELETE operations

All resources are represented in JSON format.

All resources exposed by the VXOA appliance have a common URL prefix. For example, if a resource has to be obtained from a VXOA appliance whose IP address is 1.1.1.1 and the resource URL is **/foo**, the complete URL for obtaining that resource is: http://1.1.1.1/rest/json/foo. For sake of brevity, we omit

http://1.1.1.1/rest/json from the rest of the API descriptions and refer to the resource as simply, **/foo**. The clients must use the complete URL to obtain the resource, **foo**.

Authentication

Login

Before a client can obtain or modify any resource, the client must perform login.

Create a New Authenticated HTTP Session

Use the following API to login:

```
HTTP Request:
Resource URL: /login
HTTP Method: POST
Content type: application/json
Data: {
      "user": "<username>",
      "password": "<password>"
}
--------------
HTTP Response:
HTTP code 200 – for successful login
HTTP code 401 – invalid user name or password
```

The server will return a list of cookies in the HTTP header. The 'connect.sid' cookie must be returned in every REST request after login has succeeded. For example:

```
Cookie header:
connect.sid=s%3A3%2By5e045w7vP7a7%2F1mF%2F7ylP.KprSD6kufxohhHcnbZpEvTP%2FmG%2
B%2F7U0QXO72CHBg8Fw;
```

Logout

A client can logout from a session.

Logout of Session (Denoted by the Cookie 'connect.sid' in the HTTP Header)

Logout using the following REST API:

```
HTTP Request:
Resource URL: /logout
HTTP Method: POST
```

```
Content type: application/json
Data: <empty>
--------------
HTTP Response:
HTTP code 200 – for successful logout
```

Default users

By default, the VXOA appliance has two users:

User	Password
admin	admin
monitor	monitor

The **admin** user can read-modify resources. The **monitor** user can only read resources. Without exception, all REST APIs that modify resources require **admin** privileges.

Creating new users

Creating new users requires two steps.

Get All Users Details

The client must obtain details using the following API:

```
HTTP Request:
Resource URL: /users
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "Users": {
    "admin": {
      "self": "",
      "enable": false,
      "gid": 0,
      "password": ""
    },
    "monitor": {
```

```
      "self": "",
      "enable": false,
      "gid": 0,
      "password": ""
    }
  },
  "Sessions": {
    "1": {
      "port": "",
      "gid": 0,
      "idle_time": "",
      "login_time": "",
      "remote_host": "",
      "username": ""
    },
    "2": {
      "port": "",
      "gid": 0,
      "idle_time": "",
      "login_time": "",
      "remote_host": "",
      "username": ""
    }
  }
}
```


Add / Update User Account Details

Once the client has the current list of users, the client can modify the password by setting the password key to a desired value. The client can also add new users. For example, posting the following JSON to **/users** creates two new users -- **foo** and **bar**. **foo** has administrator privileges, and **bar** has monitoring privileges. Note that the default **admin** and **monitor** users cannot be deleted.


```
{
  "users": {
    "admin": {
      "self": "admin",
      "enable": true,
      "gid": 0,
      "password": "$1$pn5x/D8c$PE4b4767D1TCrvgayHgfa/"
    },
    "monitor": {
```

```
      "self": "monitor",
      "enable": true,
      "gid": 1001,
      "password": "$1$bt53sIe.$jBu8hHD5UNgtsip.9q/GA1"
    },
    "foo": {
      "self": "foo",
      "enable": true,
      "gid": 0,
      "password": "abc"
    },
    "bar": {
      "self": "bar",
      "enable": true,
      "gid": 1001,
      "password": "def"
    }
  }
}
```


To delete a user, post the JSON list without the users that need to be deleted. Note that the "gid" field must be set to **0** for admin user and **1001** for monitor user.

RADIUS and TACACS+

To control how authentication is performed, the VXOA appliance exposes a resource named **/authentication**.

Get Current Authentication Order and Authorization Info

Clients can get this information and modify it to enable RADIUS and TACACS+ authentication. The returned data contains AAA, RADIUS, and TACACS+ objects. A typical response for this resource returns:


```
{
  "aaa": {
    "auth_method": {
      "1": {
        "self": 1,
        "name": "local"
      },
      "2": {
        "self": 2,
```

```
        "name": "radius"
      },
      "3": {
        "self": 3,
        "name": "tacacs+"
      }
    },
    "author": {
      "default-user": "admin",
      "map-order": "remote-first"
    }
  },
  "radius": {
    "server": {
      "1": {
        "auth-port": 1812,
        "address": "1.1.1.1",
        "self": 1,
        "enable": true,
        "key": "abcd",
        "retransmit": 1,
        "timeout": "3"
      }
    }
  },
  "tacacs": {
    "server": {
      "1": {
        "auth-type": "pap",
        "auth-port": 49,
        "address": "2.2.2.2",
        "self": 1,
        "enable": true,
        "key": "1234",
        "retransmit": 1,
        "timeout": "3"
      }
    }
  }
}
```

Modify Current Authentication Order and Authorization Info

The data to POST contains AAA, RADIUS, and TACACS+ objects. The “aaa” object includes “auth_method” and “author” objects.

“auth_method” represents Authentication Order and “author” represents Authorization information.

Silver Peak recommends using either RADIUS or TACACS+, but not both.

For Authentication Order, configure the following:
First = Local
Second = either RADIUS or TACACS+. If using neither, then None.
Third = None.

When using RADIUS or TACACS+ to authenticate users, configure Authorization Information as follows:
Map Order = Remote First
Default User = admin.

For radius and tacacs objects, Schema consists of a server object and a hash map of RADIUS/TACACS settings keyed by their order. You will need to construct an object of form: { 'server': { '1': {schema} }, { '2': {schema}} } for each object for POST operation to work and can configure a max of 3 RADIUS/TACACS servers. You can try GET operation for complete details of the schema.

"auth_method" object is a list of three items. This list controls the order in which authentication is performed. The "name" property can be set to "local", "radius", or "tacacs+".

"author" key controls how to determine the type of user: "admin" or "monitor". "default-user" determines the default privilege level. "map-order" can be set to "remote-first", "remote-only", or "local-only". If set to remote options, the RADIUS or TACACS+ server decides if the user has admin privileges.

If the client doesn’t have RADIUS servers, the "radius" key must be set to an empty object. Otherwise, the client can add a "server" entry to the "radius" object, as shown in the example.

If client doesn’t have TACACS+ servers, the "tacacs" key must be set to an empty object. Client can add a "server" entry to the "tacacs" object, as shown in the example. The "auth-type" field can be set to either "ascii" or "pap".

Management Interface

VXOA appliances can be configured with a varying number of interfaces to suit various network topologies. In each of these configurations, the VXOA appliance has a management interface, which is used to configure and monitor the appliance. For further details, please consult the *Silver Peak Network Deployment Guide*.

Deployment modes

1. Server mode
    a. This mode has a single interface for managing the appliance and optimizing network data.
    b. This is the default mode for virtual VXOA appliances when a virtual machine is created with a single interface.
    c. The VXOA appliance will, by default, run a DHCP client on this interface to obtain an IP address.
    d. Users can also configure the virtual machine image with a static IP address.
    e. The interface name on the VXOA appliance is **mgmt0**.
    f. This is the recommended mode for most VXOA deployments.
2. Router mode
    a. This mode has at least 3 interfaces. These are:
        i. **lan0** – data plane interface
        ii. **wan0** – data plane interface
        iii. **mgmt0** – management interface
    b. For the data plane, users can use **wan0** alone, or both the **wan0** and **lan0** interfaces.
    c. By default, the VXOA appliance runs a DHCP client on the **mgmt0** interface to obtain an IP address.
    d. Users can also configure the virtual machine image with a static IP address for the **mgmt0** interface.
3. Bridge mode
    a. This mode has at least 3 interfaces and a maximum of 6 interfaces.
        i. lan0, lan1
        ii. wan0, wan1
        iii. mgmt0, mgmt1
    b. **lan0** and **wan0** interfaces can form a bridge.
    c. The pairs, **lan0/wan0** and **lan1/wan1**, can form two distinct bridges.
    d. **mgmt0** interface configuration is the same as for server and router modes.
    e. **mgmt1** interface is used for communicating between two or more VXOA appliances. Its configuration is beyond the scope of this guide.

Getting mgmt0 address

The **mgmt0** interface is represented as resource at **URL: /managementInterface**.

The client can retrieve current management interface information using the following API:

```
HTTP Request:
Resource URL: /managementInterface
HTTP Method: GET
--------------
```

```
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "mgmt0": {
    "ipv4": {
      "dhcp": true,
      "ip": "10.0.234.210",
      "mask": 26,
      "defaultGateway": "10.0.234.193"
    }
  },
  "hostname": "silverpeak",
  "primaryDnsIP": ""
}
```

Setting mgmt0 address

Similarly, to modify properties, the client can create a response as shown above, modify the values of various properties, and perform an HTTP POST to **/managementInterface**.

If "dhcp" key is true, then the "ip", "mask", and "defaultGateway" fields are ignored.

Clients can configure the VXOA appliance hostname using this API.

Clients can configure the DNS server using this API.

> ***Note: Configuring the mgmt0 interface when a VXOA appliance has more than one interface requires many additional API requests to accomplish. Refer to the embedded REST API documentation on any 7.2 or later VXOA appliance, along with the Appliance Manager Operator's Guide for more details.***

Date and Time

VXOA appliances can be configured with time, time zone, and an NTP server to stay synchronized with a time server. It is highly recommended that users provide an NTP server to maintain the time on a VXOA appliance. Without an NTP server, the time on the appliance will drift, causing collected statistics to be inaccurate.

Getting current settings

Time information can be obtained by getting the resource at **URL: /datetime**.

```
HTTP Request:
Resource URL: /datetime
```

```
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "timeTmp": {
    "now": {
      "datetime": "2014/10/09 16:10:19",
      "date": "2014/10/09",
      "time": "16:10:19"
    },
    "zone": "UTC"
  },
  "ntp": {
    "enable": false
  },
  "currentTime": 1412871019670,
  "gmtOffset": 0
}
```

Updating current settings

To configure the time zone and NTP server information, the client needs to construct the following JSON format. The following JSON sets the time zone to UTC and the NTP server to 10.1.2.123.

```
HTTP Request:
Resource URL: /datetime
HTTP Method: POST
Content type: application/json
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON input:
{
  "timeTmp": {
    "now": {
      "datetime": "2014/10/09 16:10:19",
      "date": "2014/10/09",
      "time": "16:10:19"
    },
```

```
    "zone": "UTC"
  },
  "ntp": {
    "enable": true,
    "server": {
        "address": {
            "10.1.2.123": {
                "enable" : true,
            }
        }
    }
  }
}
```

See Appendix 1 for a complete list of time zones.

License Key

VXOA appliances of model prefix "VX" and "VRX" require a license to function. On these appliances, a license key is required to activate the appliance.

Getting appliance's key and validity

License information can be obtained by getting the license resource at **URL: /license**.

```
HTTP Request:
Resource URL: /license
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "licenseKey": "abcdefg",
  "startDate": "2014/07/08 12:00:00",
  "endDate": "2014/10/06 12:00:00",
  "serial": "00-11-22-33-44-55",
  "model": "VX5000",
  "display": "VX-5000"
}
```

Setting the key

License keys can be set by posting new license information to the same URL.

```
HTTP Request:
Resource URL: /license
HTTP Method: POST
Content type: application/json
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON input:
{
  "licenseKey": { "abcd" }
}
```

Checking if reboot required

If it's the first time applying a license, the appliance may require a reboot. To determine if a reboot is required, the client must get a resource at **URL: /systemInfo**. Look at the highlighted field in the systemInfo API description to determine if a reboot is required after applying the license.

System Information

System information can be obtained using a read-only resource at **URL: /systemInfo**. It provides a list of system level properties of a VXOA appliance.

```
HTTP Request:
Resource URL: /systemInfo
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "hostName": "dall",
  "applianceid": 65852,
  "model": "NX-8600 200181-001 Rev 2",
  "modelShort": "NX8600",
  "status": "Normal",
  "uptime": "472476096",
  "uptimeString": "5d 11h 14m 36s",
  "datetime": "2014/10/09 16:46:09 Etc/UTC",
  "gmtOffset": 0,
  "release": "VXOA 0.0.0.0_53311p",
```

```
  "releaseWithoutPrefix": "0.0.0.0_53311p",
  "serial": "00-1B-BC-01-01-3C",
  "uuid": "425cf5b9-889f-414a-ae78-1cf07faa2675",
  "deploymentMode": "router",
  "licenseRequired": false,
  "isLicenseInstalled": false,
  "licenseExpiryDate": "",
  "licenseExpirationDaysLeft": 1.7976931348623157e+308,
  "hasUnsavedChanges": true,
  "rebootRequired": false,
  "alarmSummary": {
    "num_critical": 2,
    "num_outstanding": 2,
    "num_equipment_outstanding": 0,
    "num_major": 0,
    "num_tca_outstanding": 0,
    "num_minor": 0,
    "num_software_outstanding": 0,
    "num_traffic_class_outstanding": 0,
    "num_tunnel_outstanding": 2,
    "num_warning": 0
  }
}
```


Saving Changes

All REST requests that modify the state on the VXOA appliance only make changes to the in-memory state. To make changes persist across reboots, clients must call a REST API to save all the changes to disk. It can be done by HTTP POST to **URL: /saveChanges**.

```
HTTP Request:
Resource URL: /saveChanges
HTTP Method: POST
Content type: application/json
--------------
HTTP Response:
HTTP code 200 – for successful response
HTTP Body: <empty>
```

Rebooting the Appliance

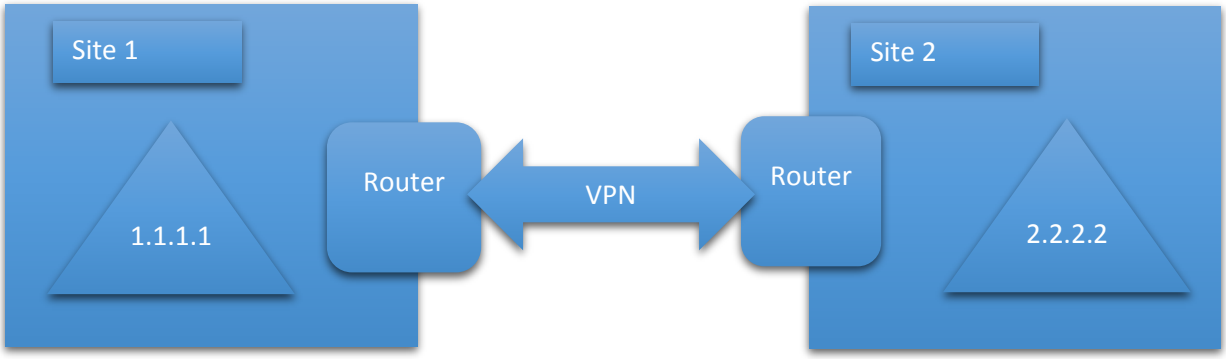
A VXOA appliance can be rebooted by performing HTTP POST to **URL: /reboot**.

```
HTTP Request:
Resource URL: /reboot
HTTP Method: POST
Content type: application/json
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON input:
{
  "reboot_type" : "Normal",
  "save_db" : true
}
Reboot type can be one of following: Normal, Forced, Shutdown. If a VXOA
appliance is shutdown, user will have to manually power up the appliance in
order to gain communication to the appliance.
"save_db" is another way to both save changes and reboot the appliance at the
same time instead of saving changes by doing an HTTP POST to /saveChanges.
```

Tunnels

Two Silver Peak VXOA appliances work in tandem to perform various optimizations over a Wide Area Network. In many deployments, VXOA appliances automatically establish tunnels, based on the traffic patterns they see. Under certain network configurations, tunnels may not be set up automatically. In these situations, tunnels must be configured manually.

The following diagram illustrates the use of tunnel APIs. In this example, two VXOA appliances, whose IPs are 1.1.1.1 and 2.2.2.2, are in Server Mode with a single interface. The WAN routers have established a VPN allowing networks at two sites to communicate directly, without any sort of NAT.



Getting current list of tunnels and config, uptime, state info

The current list of tunnels can be obtained by getting a resource at **URL: /tunnelsConfigAndState**.
Sample response format:

```
HTTP Request:
Resource URL: /tunnelsConfigAndState
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "tunnel1": {
    "max_bw_auto": true,
    "max_bw": 500000,
    "admin": "up",
    "mode": "udp",
    "self": "tunnel1",
    "destination": "2.2.2.2",
    "auto_mtu": true,
    "source": "1.1.1.1",
    "ipsec_enable": false,
    "mtu": 1500,
    "status": "Up - Active",
    "presharedkey": "ipsec key"
  }
}
```

Some of the properties of a tunnel resource are explained below:

1. **admin** – the administrative state of the tunnel. It takes two values: "up" or "down".
2. **mode** – the encapsulation used on packets sent through this tunnel. This can be set to "udp", "gre", or "ipsec".
3. **destination** – IP address of the remote VXOA appliance.
4. **source** – IP address of the local VXOA appliance.
5. **max_bw_auto** – if *true*, this sets the tunnel bandwidth to a number that can vary up to a maximum of the system bandwidth value. Silver Peak recommends setting this value to *true*.
6. **max_bw** – for setting a specific maximum bandwidth value for a tunnel, in Kbps.
7. **auto_mtu** – tunnels can discover the end-to-end MTU value that's safe to use between two VXOA appliances. Setting this value to *true* is recommended.
8. **mtu** – if auto_mtu is set to *false*, this value represents the manually configured MTU for the tunnel.

9. **presharedkey** – if tunnel mode is set to "ipsec", this value is the pre-shared key for the IPSec protocol.
10. **status** – this represents the operational status of a tunnel.

Adding / deleting / modifying tunnels

To create tunnels on the two VXOA appliances in the example above, the client must perform HTTP POST operations on both of them. The tunnels can be created or modified by performing an HTTP POST operation on the **URL: /tunnels**.

> ***NOTE: When creating or modifying multiple tunnels, clients must GET the entire tunnel list from the appliance and then add, modify, or delete the entries in the list, and POST the entire list back to the appliance.***

Example – Tunnel Creation between Two Sites (Site 1 and Site 2)

Site 1:

Here we create a tunnel to Site 2 with UDP encapsulation. MTU and Bandwidth are both set to "Auto".

```
HTTP Request:
Resource URL: /tunnels
HTTP Method: POST
Content type: application/json
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "tunnel_from_site1_to_site2": {
    "max_bw_auto": true,
    "admin": "up",
    "mode": "udp",
    "self": " tunnel_from_site1_to_site2",
    "destination": "2.2.2.2",
    "auto_mtu": true,
    "source": "1.1.1.1",
  }
}
```

Site 2:

Here we create a tunnel to Site 1 with UDP encapsulation. MTU is set to 1200, and Bandwidth is set to 10,000 Kbps.

```
HTTP Request:
Resource URL: /tunnels
HTTP Method: POST
Content type: application/json
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "tunnel_from_site2_to_site1": {
    "max_bw_auto": false,
    "max_bw": 10000,
    "admin": "up",
    "mode": "udp",
    "self": "tunnel_from_site2_to_site1",
    "destination": "1.1.1.1",
    "auto_mtu": false,
    "mtu": 1200,
    "source": "2.2.2.2",
  }
}
```

Disabling auto-tunnel

As mentioned in the previous section, VXOA appliances can automatically discover traffic patterns and create tunnels. If such behavior is not desirable, disable this feature by performing HTTP POST to **URL: /system**.

```
HTTP Request:
Resource URL: /system
HTTP Method: POST
Content type: application/json
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "auto_tunnel": false
}
```

NOTE: When creating or modifying multiple tunnels, clients must GET the entire tunnel list from the appliance and then add, modify, or delete the entries in the list, and POST the entire list back to the appliance.

Traffic Optimization

VXOA appliances can optimize network traffic by redirecting flows through the tunnels created in the previous section.

Traffic can be directed into a specific tunnel by using three different techniques:

1. Auto-optimization

   VXOA appliances can automatically resolve which destination tunnel to use if network traffic flows symmetrically between the two appliances. If this type of network topology can be designed, no configuration is necessary to optimize traffic.

2. Subnet sharing

   Two VXOA appliances can exchange information about the subnets on their respective sites. This allows them to redirect traffic destined to a subnet, then to a specific tunnel, then to a VXOA appliance. This type of optimization requires some configuration.

3. Route Policies

   Users can manually create access control lists to route packets to various destinations through specific tunnels. This requires more detailed configuration.

The following sections describe how to configure subnet sharing and route policies. For a detailed description of how these features work, consult the *Appliance Manager Operator's Guide*.

Subnet sharing – enabling, disabling, adding local

The JSON format for configuring the subnet sharing feature is described below.

To enable the subnet sharing feature on an appliance, the client must perform an HTTP POST to **URL: /system**.

1. "self" must be set to *true* to enable the feature. Setting it to *false* disables the feature.
2. "add_local" allows the VXOA appliance to **automatically** share the subnets on its local interfaces. By default, the metric for automatically shared subnets is set to 50.

```
HTTP Request:
```

```
Resource URL: /system
HTTP Method: POST
Content type: application/json
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "auto_subnet": {
      "self" : true,
      "add_local" : true,
      "add_local_metric": 50
  }
}
```

Adding configured local subnets

To add specific subnets that are local to this site and need to be shared with remote VXOA appliances, clients must perform HTTP POST to **URL: /subnets/configured**.

The format of the JSON data to be posted is explained below. The data to be posted is a list of objects whose key is the subnet to be advertised to the other VXOA appliances. Each subnet has four properties:

1. **local** – if *true*, it indicates that the subnet is truly local. Some sites may want to advertise subnets that may not be local, just to get all the traffic to that site for security or other reasons.
2. **advert** – if *true*, the subnet is advertised to other VXOA appliances.
3. **exclude** – this option is used to exclude more specific subnets if a larger subnet is being advertised.
4. **metric** – this is a number between 0 and 100. Lower values indicate higher priority.

```
HTTP Request:
Resource URL: /subnets/configured
HTTP Method: POST
Content type: application/json
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
"ipv4":{
        "3.3.3.3/32":{
            "local":true,
            "advert":true,
```

```
            "exclude":false,
            "metric":50}}}
```

Getting configured and learned subnets, automatically learned local subnets

To view a list of configured and learned subnets on a VXOA appliance, clients can perform an HTTP GET operation on **URL: /subnets/all**.

```
HTTP Request:
Resource URL: /subnets/all
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "subnets": {
    "count": 2,
    "max": 4000,
    "entries": [
      {
        "prefix": "157.10.2.0/23",
        "metric": 50,
        "peerid": 65852,
        "learned": false,
        "configured": false,
        "advert": true,
        "automatic": true,
        "exclude": false,
        "local": true,
        "learnedFromIp": "",
      },
      {
        "prefix": "3.3.3.3/32",
        "metric": 50,
        "peerid": 65852,
        "learned": false,
        "configured": true,
        "advert": true,
        "automatic": false,
        "exclude": false,
        "local": true,
```

```
        "learnedFromIp": "",
      }
    ]
  }}
```

Route policies

Route policies are required when network topology makes automatic optimization impractical. Route policies are stored in containers called "maps". An appliance can have many maps – but only one of them can be active at any given time. A map is a list of rules, with each rule having a priority. A lower number for priority indicates a higher precedence. The priorities take values from 1 to 65534.

Each rule has some fields which are used for matching attributes in a network packet.

Match fields

1. Protocol
    a. Most common values for protocol field are **ip**, **tcp**, and **udp**.
    b. You can use any integer from 0 to 255. This represents the protocol field in the IP Header of the packet.
2. Source subnet
    a. It can be a specific IP address, or an IP subnet in the form, x.x.x.x/x.
    b. To match any IPv4 addresss, use 0.0.0.0/0
    c. To match any IPv6 address, use ::0/0
3. Destination subnet
    a. It can be a specific IP address, or an IP subnet in the form, x.x.x.x/x.
    b. To match any IPv4 addresss, use 0.0.0.0/0
    c. To match any IPv6 address, use ::0/0
4. Application
    a. Users can also define applications based on TCP and UDP ports and refer to the application by name in route policies. If you need to define applications using a REST API, contact Silver Peak for more information.
5. Source TCP/UDP Port
6. Destination TCP/UDP Port
7. DSCP marking on the packet
8. VLAN marking on the packet

Each rule also has an action which gets executed if all the properties specified in the match section are satisfied. Each action has following fields:

1. **Destination**: auto, pass-through, pass-through-unshaped, drop, a tunnel name, or a VXOA appliance hostname
2. **Path**: default, load-balance, low-loss, low-latency, a specific interface
3. **Fallback**: pass-through, pass-through-unshaped, drop, continue

Below is an example of a top-level route policy container. The name of the container in this example is "map1". The "options" object contains information about which container is currently active.

```
{
    "data":{
        "map1":{ }
    },
    "options":{
        "activeMap":"map1"
    }
}
```

"map1" contains a number of rules and each rule has a priority. Its JSON structure is shown below:

```
"map1":{
            "prio":{
                "1":{ },
                "10":{ },
                "65535":{ }
            }
        }
```

Each rule has the following JSON structure:

```
"1":{
                    "set":{ },
                    "match":{ }
                }
```

The "match" section has the following JSON structure:

- You can see that in the "dest" and "source" sections, there are distinct keys for IPv4 and IPv6 subnets. Both source and destination sections must be either IPv4 or IPv6.
- The "ipver" field must be either "ipv4" or "ipv6".
- The DSCP field can take values from 0 to 63.
- The VLAN field can take values from 1 to 4094.
- The "protocol" field can be "ip", "tcp", or "udp". It can also be a number from 0 to 255.

```
"match": {
            "0": {
              "dest": {
                "addr": {
                  "ipv4": {
                    "static": {
```

```
                    "mask_len": "255.255.255.0",
                    "ipaddr": "2.2.2.0"
                  }
                },
                "ipv6": {
                  "static": {
                    "ipv6prefix": "::/0"
                  }
                }
              },
              "port": 80
            },
            "self": 0,
            "dscp": "12",
            "protocol": "tcp",
            "ipver": "ipv4",
            "application": "any",
            "source": {
              "addr": {
                "ipv4": {
                  "static": {
                    "mask_len": "255.255.255.0",
                    "ipaddr": "1.1.1.0"
                  }
                },
                "ipv6": {
                  "static": {
                    "ipv6prefix": "::/0"
                  }
                }
              }
            },
            "src": {
              "port": 1234
            },
            "vlan": "any.10"
          }
        },
```


The "set" JSON object has the following format:

```
"set": {
            "0": {
              "auto_mod": "default",
              "peer": "",
              "auto_ena": false,
              "drop_ena": false,
              "auto_prefif": "none",
              "down_action": "pass-through",
              "passthru_ena": false,
              "passthru_shape_ena": true,
              "tun_pri": "tunnel_name"
            }
          }
```

For the **set** action, the client must set the fields appropriately for different types of actions. The following table provides an easy guide for configuring set actions. An empty cell indicates that the value of the cell is the same as the cell value in the previous row.

Type of action	auto_mod	peer	auto_ena	drop_ena	auto_prefif	down_action	passthru_ena	passthru_shape_ena	tun_pri
Auto-optimize	"default"	""	true	false	"none"	"pass-through" or "pass-through-unshaped" or "drop"	false	true	""
Pass-through shaped			false				true		
Pass-through Unshaped								false	
Drop				true			false		
Tunnel named "t1"				false					"t1"

Peer appliance named "foo"	"balance" or "low-loss" or "low-latency"	"foo"							""
Use a specific interface e.g. wan0	"preferred-if"	""	true		"wan0"				

To post a set of route policies, the JSON structure looks like this:

```
{
    "data":{
        "map1":{ }
    },
    "options":{
        "activeMap":"map1",
        "merge":false,
        "templateApply":false
    }
}
```

The contents of map1 are as described in previous pages. The contents need to be posted to **URL: /routeMaps**.

Getting and modifying remote syslog receivers

VXOA appliances can send log entries to a syslog server.

Clients can retrieve and modify syslog configuration by accessing a resource at **URL: /logging/remote**.

```
HTTP Request:
Resource URL: /logging/remote
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
 [
  {
    "ip": "1.1.1.1",
    "self": {
      "fac": "all",
      "min_severity": "Error"
    }
  }
]
```

To modify, delete, or add new syslog receivers, use the same JSON data format to create an HTTP POST to **URL: /logging/remote**.

The "min_severity" field can take the following options:
None, Emergency, Alert, Critical, Error, Warning, Notice, Info, and Debug

Enabling, setting, getting SNMP server settings

VXOA appliances support read-only SNMP access of standard MIB-II objects and Silver Peak–specific objects. You can report alarms to an SNMP receiver by using SNMP traps.

SNMP configuration can be retrieved by accessing resource at **URL: /snmp**.

```
HTTP Request:
Resource URL: /snmp
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
    "access":{
        "rocommunity":"public"
    },
    "listen":{
        "enable":true
    },
    "traps":{
        "trap_community":"public",
        "enable":true
    },
    "auto_launch":true,
    "sysdescr":"",
    "syscontact":"",
    "syslocation":"",
    "v3":{
        "users":{
            "admin":{
                "hash_type":"sha",
                "auth_key":"",
                "self":"admin",
                "privacy_key":"",
                "privacy_type":"aes-128",
                "enable":false
            }
        }
    },
    "encAuth":false,
```

```
    "encPri":false,
    "trapsink":{
        "sink":{
            "2.2.2.2":{
                "self":"2.2.2.2",
                "enable":true,
                "community":"abcd",
                "type":"trap-v1"
            }
        }
    }
}
```

- To enable SNMP, set "auto_launch" to *true* and "listen.enable" to *true*.
- To enable traps, set "traps.trap_community" to a valid string and "traps.enable" to *true*.
- To add to a list of trap receivers, provide a trap object as shown in the example. These objects need to be added under the "trapsink.sink" object.

Trap types can be either trap-v1 or trap-v2c.

By default, there is a disabled, built-in admin user for SNMPv3.
To enable the SNMPv3 admin user:

- Set **v3.users.admin.enable flag** to *true*.
- Set **v3.users.admin.auth_key** and **v3.users.admin.privacy_key** to whatever values you choose.
- "hash_type" can be "sha" or "md5".
- "privacy_type" can be either "aes-128" or "des".

To obtain the SNMP Enterprise MIB for VXOA appliances, you can either contact Silver Peak Support or download the MIBS and a companion document from the Support > User Documentation webpage.

Monitoring the VXOA Appliance

VXOA appliances can be monitored using REST APIs. Each type of data exposed by the appliance is represented as a resource in JSON format and can be obtained by using the HTTP GET operation from a specific URL. The following sections describe some of the commonly used monitoring resources.

Alarms

Alarm information can be received by performing HTTP GET on **URL: /alarms**. Alarm information has following JSON structure. Severity can be mapped as follows:

1. 4 -> Critical
2. 3 -> Major
3. 2 -> Minor
4. 1 -> Warning

```
HTTP Request:
Resource URL: /alarms
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 - for successful response
Sample JSON output:
{
  "outstanding": {
    "1": {
      "alarm_string": "Fri Oct  3 22:33:11 2014
,0x10008,CRI,N,dall_157.10.2.11_to_falcon_157.10.1.11,Tunnel local IP address
not owned by this appliance",
      "active": true,
      "severity": 4,
      "service_affect": true,
      "sequence_id": 5,
      "source": "dall_157.10.2.11_to_falcon_157.10.1.11",
      "occurence_count": 1,
      "self": 1,
      "acknowledged": false,
      "time": "05:33:11",
      "clearable": false,
      "description": "Tunnel local IP address not owned by this appliance",
      "name": "tunnel_id_bad_local_addr",
      "type": "TUN",
      "type_id": 65544
    },
    "2": {
      "alarm_string": "Fri Oct  3 22:33:11 2014
,0x10008,CRI,N,dall_157.10.2.11_to_falcon_157.10.1.10,Tunnel local IP address
not owned by this appliance",
      "active": true,
      "severity": 4,
      "service_affect": true,
      "sequence_id": 4,
      "source": "dall_157.10.2.11_to_falcon_157.10.1.10",
      "occurence_count": 1,
      "self": 2,
      "acknowledged": false,
      "time": "05:33:11",
      "clearable": false,
      "description": "Tunnel local IP address not owned by this appliance",
```

```
      "name": "tunnel_id_bad_local_addr",
      "type": "TUN",
      "type_id": 65544
    }
  },
  "summary": {
    "num_critical": 2,
    "num_outstanding": 2,
    "num_equipment_outstanding": 0,
    "num_major": 0,
    "num_tca_outstanding": 0,
    "num_minor": 0,
    "num_software_outstanding": 0,
    "num_traffic_class_outstanding": 0,
    "num_tunnel_outstanding": 2,
    "num_warning": 0
  }
}
```


Real-time statistics

VXOA appliances collect various statistics on a per-second basis. They maintain a buffer of 3 seconds for these statistics. The clients must poll at an interval of less than three seconds to make sure that they don't lose any samples.

Real time statistics are obtained using following REST API. The URL is **/stats/realtimeStats**.

```
HTTP Request:
Resource URL: /stats/realtimeStats
HTTP Method: POST
Content type: application/json
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON input:
{
   filter: "0",
   name: "foo",
   "type": "bar"
}
```

The following table shows various settings for filter, name, and type keys used to retrieve different types of real time statistics. Filter can take following values:

- "0" for traffic across all tunnels on the VXOA appliance.
- "1" for traffic going pass-through shaped.
- "2" for traffic going pass-through unshaped.
- "3" for all traffic through the VXOA appliance.

Type of statistics	Filter	Name	Type
Statistics on a specific tunnel named "xxx"	This argument is ignored.	"xxx"	"tunnel"
Statistics for traffic categories	0 or 3	"0"	"trafficType"
DSCP statistics	0 or 3	0 to 63 (different DSCP markings)	"dscp"
Traffic Class statistics	0 or 3	"0" to "9" (different traffic classes)	"trafficClass"
Flow statistics	0, 1, 2, or 3	0 for TCP accelerated flow counts 1 for TCP unaccelerated flow counts 2 for non-TCP flow counts	"flow"
Application statistics	0, 1, 2, or 3	Name of the application as defined using user-defined applications APIs.	"app"

Results for these different statistics are very similar and self-explanatory. Here is an example result for tunnel statistics:

```
{
  "name of metric": [
    [
      <unix_time_in_microseconds>,
      <value of metric>
    ],
    [
      <unix_time_in_microseconds>,
      <value of metric>
    ],
    [
      <unix_time_in_microseconds>,
      <value of metric>
    ],
  ]
}
```

Each metric is an array of three items. Each item contains a Unix timestamp in microseconds and the value of the metric. The three items are one second apart in interval. The metric values are not cumulative.

Most metrics have following common prefixes:

1. **WRX** for WAN Receive and **WTX** for WAN Transmit
2. **LRX** for LAN Receive and **LTX** for LAN Transmit

The following metrics are reported for different categories:

1. WTX_BYTES, WRX_BYTES, LRX_BYTES, LTX_BYTES:
   bytes transmitted or received during the last second
2. WTX_PKTS, WRX_PKTS, LTX_PKTS, LRX_PKTS:
   packets transmitted or received during the last second
3. COMP_L2W and COMPW2L:
   Compression in LAN-to-WAN direction and compression in WAN-to-LAN direction, respectively
4. LATENCY_AVG, LATENCY_MIN
5. TCP_FLOWS, TCP_ACC_FLOWS, NON_TCP_FLOWS
6. CREATED, DELETED:
   Flows created and deleted during last second
7. PRE_PCT_LOSS, POST_PCT_LOSS:
   Loss before Forward Error Correction (FEC) and Loss after FEC.
8. PRE_PCT_POC, POST_PCT_POC:
   Out-of-order packets before and after correction

Historical statistics

VXOA appliances store historical statistics at the one-minute interval for three days. To obtain these statistics, clients can poll at intervals of one minute or larger. The URL for getting these statistics is:

/stats/minuteStats/json/:category/:minute

category can take following values:

1. flow
2. dscp
3. trafficclass
4. application
5. tunnel

The minute parameter is the exact minute boundary of time represented as Unix time in seconds.

To retrieve the minute range for which the appliance has statistics, clients can perform an HTTP GET request on **URL: /stats/minuteRange**.

```
HTTP Request:
Resource URL: /stats/minuteRange
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "newest": "1412894100",
  "oldest": "1412634660"
}
```

These timestamps can be used to retrieve statistics for specific minutes between "newest" and "oldest" timestamps.

For example, an HTTP GET request on **URL: /stats/minuteStats/json/tunnel/1412894100** may return a JSON object like following:

```
HTTP Request:
Resource URL: /stats/minuteRange
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "tunnel_to_new_york": {
    "wtx_bytes": "0",
    "wrx_bytes": "0",
    "ltx_bytes": "0",
    "lrx_bytes": "0",
    "wtx_pkts": "0",
    "wrx_pkts": "0",
    "ltx_pkts": "0",
    "lrx_pkts": "0",
    "latency": "80",
    "pre_fec_wrx_packet_loss": "0",
    "post_fec_wrx_packet_loss": "0",
    "pre_poc_wrx_packet_ooo": "0",
    "post_poc_wrx_packet_ooo": "0"
```

```
  }
}
```

Image Install and Upgrade

VXOA appliances can be upgraded using REST APIs.

Get current versions on both partitions

Current version information can be obtained by performing an HTTP GET at **URL: /softwareVersions**. Each VXOA appliance has two image partitions. Each partition holds a software version. There is an active partition which holds currently running software. The "active" flag indicates which partition is currently active. The "next_boot" flag indicates which partition will be used when the VXOA appliance is rebooted.

```
HTTP Request:
Resource URL: /softwareVersions
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
[
  {
    "partition": 1,
    "build_version": "0.0.0.0_53146p",
    "build_time": "2014-09-25 12:33:27",
    "active": false,
    "next_boot": false,
    "fallback_boot": true
  },
  {
    "partition": 2,
    "build_version": "0.0.0.0_53311p",
    "build_time": "2014-10-03 22:13:55",
    "active": true,
    "next_boot": true,
    "fallback_boot": false
  }
]
```

Install software image

To upgrade software on a VXOA appliance, the client must obtain a software image and make it accessible, using HTTP protocol. The REST API to upgrade software requires the HTTP URL of the image. A software upgrade can be initiated by performing an HTTP POST to **URL: /install**.

```
HTTP Request:
Resource URL: /install
HTTP Method: POST
Content type: application/json
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
    "downloadOption":"url",
    "installOption":"install_only",
    "options":{
        "image_url":"http://mi3/share/builds/latest/image-7.1.0.0_53372.img"
    }
}
```

"installOption" can take the following values:

1. **install_only** – installs the image in the inactive partition.
2. **install_reboot** – installs the image in the inactive partition and upon successful installation, reboots from the partition having the newly installed image.
3. **install_switch** – installs the image in the inactive partition and sets the "next_boot" flag to *true* for that partition. During the next reboot, the VXOA appliance loads the new software image.

Check status of upgrade process

To determine if the software upgrade process is complete and successful, the client must perform an HTTP GET operation to the **URL: /upgradeSoftware/status**.

```
HTTP Request:
Resource URL: /upgradeSoftware/status
HTTP Method: GET
--------------
HTTP Response:
HTTP code 200 – for successful response
Sample JSON output:
{
  "current_bytes": 1745026,
  "transfer_rate": 2792041,
```

```
  "download_end": 0,
  "install_start": 0,
  "install_end": 0,
  "state": 2,
  "download_start": 1412896202,
  "message": "Downloading the boot image file",
  "total_bytes": 149574899,
  "percent_complete": 1
}
```


- When no install is in progress, "state" is set to "1".
- When a download is successfully started, "state" is set to "2" and "percent_complete" shows the completion progress of the download.
- After the download is complete, actual installation starts and "state" is set to "3". Once again, "progress_complete" starts from "0" and reflects the installation progress.
- Once installation is complete, "state" is set back to "1".
- To test for completion, you can also use "download_end" and "install_end". They are set to "0" at the start of the upgrade process.

Appendix 1

Time zones

"Africa/Abidjan",
"Africa/Accra",
"Africa/Addis Ababa",
"Africa/Algiers",
"Africa/Asmera",
"Africa/Bamako",
"Africa/Bangui",
"Africa/Banjul",
"Africa/Bissau",
"Africa/Blantyre",
"Africa/Brazzaville",
"Africa/Bujumbura",
"Africa/Cairo",
"Africa/Casablanca",
"Africa/Ceuta",
"Africa/Conakry",
"Africa/Dakar",
"Africa/Dar es Salaam",
"Africa/Djibouti",
"Africa/Douala",
"Africa/El Aaiun",
"Africa/Freetown",
"Africa/Gaborone",
"Africa/Harare",
"Africa/Johannesburg",
"Africa/Kampala",
"Africa/Khartoum",
"Africa/Kigali",
"Africa/Kinshasa",
"Africa/Lagos",
"Africa/Libreville",
"Africa/Lome",
"Africa/Luanda",
"Africa/Lubumbashi",
"Africa/Lusaka",
"Africa/Malabo",
"Africa/Maputo",
"Africa/Maseru",
"Africa/Mbabane",
"Africa/Mogadishu",
"Africa/Monrovia",
"Africa/Nairobi",
"Africa/Ndjamena",
"Africa/Niamey",

```
"Africa/Nouakchott",
"Africa/Ouagadougou",
"Africa/Porto-Novo",
"Africa/Sao Tome",
"Africa/Timbuktu",
"Africa/Tripoli",
"Africa/Tunis",
"Africa/Windhoek",
"America/Adak",
"America/Anchorage",
"America/Anguilla",
"America/Antigua",
"America/Araguaina",
"America/Aruba",
"America/Asuncion",
"America/Atka",
"America/Barbados",
"America/Belem",
"America/Belize",
"America/Boa Vista",
"America/Bogota",
"America/Boise",
"America/Buenos Aires",
"America/Cambridge Bay",
"America/Cancun",
"America/Caracas",
"America/Catamarca",
"America/Cayenne",
"America/Cayman",
"America/Chicago",
"America/Chihuahua",
"America/Cordoba",
"America/Costa Rica",
"America/Cuiaba",
"America/Curacao",
"America/Danmarkshavn",
"America/Dawson",
"America/Dawson Creek",
"America/Denver",
"America/Detroit",
"America/Dominica",
"America/Edmonton",
"America/Eirunepe",
"America/El Salvador",
"America/Ensenada",
"America/Fort Wayne",
"America/Fortaleza",
"America/Glace Bay",
```

```
"America/Godthab",
"America/Goose Bay",
"America/Grand Turk",
"America/Grenada",
"America/Guadeloupe",
"America/Guatemala",
"America/Guayaquil",
"America/Guyana",
"America/Halifax",
"America/Havana",
"America/Hermosillo",
"America/Indiana/Indianapolis",
"America/Indiana/Knox",
"America/Indiana/Marengo",
"America/Indiana/Vevay",
"America/Indianapolis",
"America/Inuvik",
"America/Iqaluit",
"America/Jamaica",
"America/Jujuy",
"America/Juneau",
"America/Kentucky/Louisville",
"America/Kentucky/Monticello",
"America/Knox IN",
"America/La Paz",
"America/Lima",
"America/Los Angeles",
"America/Louisville",
"America/Maceio",
"America/Managua",
"America/Manaus",
"America/Martinique",
"America/Mazatlan",
"America/Mendoza",
"America/Menominee",
"America/Merida",
"America/Mexico City",
"America/Miquelon",
"America/Monterrey",
"America/Montevideo",
"America/Montreal",
"America/Montserrat",
"America/Nassau",
"America/New York",
"America/Nipigon",
"America/Nome",
"America/Noronha",
"America/North_Dakota/Center",
```

```
"America/Panama",
"America/Pangnirtung",
"America/Paramaribo",
"America/Phoenix",
"America/Port-au-Prince",
"America/Port of Spain",
"America/Porto Acre",
"America/Porto Velho",
"America/Puerto Rico",
"America/Rainy River",
"America/Rankin Inlet",
"America/Recife",
"America/Regina",
"America/Rio Branco",
"America/Rosario",
"America/Santiago",
"America/Santo Domingo",
"America/Sao Paulo",
"America/Scoresbysund",
"America/Shiprock",
"America/St Johns",
"America/St Kitts",
"America/St Lucia",
"America/St Thomas",
"America/St Vincent",
"America/Swift Current",
"America/Tegucigalpa",
"America/Thule",
"America/Thunder Bay",
"America/Tijuana",
"America/Tortola",
"America/Vancouver",
"America/Virgin",
"America/Whitehorse",
"America/Winnipeg",
"America/Yakutat",
"America/Yellowknife",
"Antarctica/Casey",
"Antarctica/Davis",
"Antarctica/DumontDUrville",
"Antarctica/Mawson",
"Antarctica/McMurdo",
"Antarctica/Palmer",
"Antarctica/Rothera",
"Antarctica/South Pole",
"Antarctica/Syowa",
"Antarctica/Vostok",
"Arctic/Longyearbyen",
```

```
"Asia/Aden",
"Asia/Almaty",
"Asia/Amman",
"Asia/Anadyr",
"Asia/Aqtau",
"Asia/Aqtobe",
"Asia/Ashgabat",
"Asia/Ashkhabad",
"Asia/Baghdad",
"Asia/Bahrain",
"Asia/Baku",
"Asia/Bangkok",
"Asia/Beirut",
"Asia/Bishkek",
"Asia/Brunei",
"Asia/Calcutta",
"Asia/Choibalsan",
"Asia/Chongqing",
"Asia/Chungking",
"Asia/Colombo",
"Asia/Dacca",
"Asia/Damascus",
"Asia/Dhaka",
"Asia/Dili",
"Asia/Dubai",
"Asia/Dushanbe",
"Asia/Gaza",
"Asia/Harbin",
"Asia/Hong Kong",
"Asia/Hovd",
"Asia/Irkutsk",
"Asia/Istanbul",
"Asia/Jakarta",
"Asia/Jayapura",
"Asia/Jerusalem",
"Asia/Kabul",
"Asia/Kamchatka",
"Asia/Karachi",
"Asia/Kashgar",
"Asia/Katmandu",
"Asia/Krasnoyarsk",
"Asia/Kuala Lumpur",
"Asia/Kuching",
"Asia/Kuwait",
"Asia/Macao",
"Asia/Macau",
"Asia/Magadan",
"Asia/Makassar",
```

"Asia/Manila",
"Asia/Muscat",
"Asia/Nicosia",
"Asia/Novosibirsk",
"Asia/Omsk",
"Asia/Oral",
"Asia/Phnom_Penh",
"Asia/Pontianak",
"Asia/Pyongyang",
"Asia/Qatar",
"Asia/Qyzylorda",
"Asia/Rangoon",
"Asia/Riyadh",
"Asia/Riyadh87",
"Asia/Riyadh88",
"Asia/Riyadh89",
"Asia/Saigon",
"Asia/Sakhalin",
"Asia/Samarkand",
"Asia/Seoul",
"Asia/Shanghai",
"Asia/Singapore",
"Asia/Taipei",
"Asia/Tashkent",
"Asia/Tbilisi",
"Asia/Tehran",
"Asia/Tel Aviv",
"Asia/Thimbu",
"Asia/Thimphu",
"Asia/Tokyo",
"Asia/Ujung Pandang",
"Asia/Ulaanbaatar",
"Asia/Ulan Bator",
"Asia/Urumqi",
"Asia/Vientiane",
"Asia/Vladivostok",
"Asia/Yakutsk",
"Asia/Yekaterinburg",
"Asia/Yerevan",
"Atlantic/Azores",
"Atlantic/Bermuda",
"Atlantic/Canary",
"Atlantic/Cape_Verde",
"Atlantic/Faeroe",
"Atlantic/Jan Mayen",
"Atlantic/Madeira",
"Atlantic/Reykjavik",
"Atlantic/South Georgia",

```
"Atlantic/St Helena",
"Atlantic/Stanley",
"Australia/ACT",
"Australia/Adelaide",
"Australia/Brisbane",
"Australia/Broken Hill",
"Australia/Canberra",
"Australia/Darwin",
"Australia/Hobart",
"Australia/LHI",
"Australia/Lindeman",
"Australia/Lord Howe",
"Australia/Melbourne",
"Australia/NSW",
"Australia/North",
"Australia/Perth",
"Australia/Queensland",
"Australia/South",
"Australia/Sydney",
"Australia/Tasmania",
"Australia/Victoria",
"Australia/West",
"Australia/Yancowinna",
"Brazil/Acre",
"Brazil/DeNoronha",
"Brazil/East",
"Brazil/West",
"Chile/Continental",
"Chile/EasterIsland",
"Canada/Atlantic",
"Canada/Central",
"Canada/East-Saskatchewan",
"Canada/Eastern",
"Canada/Mountain",
"Canada/Newfoundland",
"Canada/Pacific",
"Canada/Saskatchewan",
"Canada/Yukon",
"Cuba",
"Egypt",
"Europe/Amsterdam",
"Europe/Andorra",
"Europe/Athens",
"Europe/Belfast",
"Europe/Belgrade",
"Europe/Berlin",
"Europe/Bratislava",
"Europe/Brussels",
```

```
"Europe/Bucharest",
"Europe/Budapest",
"Europe/Chisinau",
"Europe/Copenhagen",
"Europe/Dublin",
"Europe/Gibraltar",
"Europe/Helsinki",
"Europe/Istanbul",
"Europe/Kaliningrad",
"Europe/Kiev",
"Europe/Lisbon",
"Europe/Ljubljana",
"Europe/London",
"Europe/Luxembourg",
"Europe/Madrid",
"Europe/Malta",
"Europe/Minsk",
"Europe/Monaco",
"Europe/Moscow",
"Europe/Nicosia",
"Europe/Oslo",
"Europe/Paris",
"Europe/Prague",
"Europe/Riga",
"Europe/Rome",
"Europe/Samara",
"Europe/San Marino",
"Europe/Sarajevo",
"Europe/Simferopol",
"Europe/Skopje",
"Europe/Sofia",
"Europe/Stockholm",
"Europe/Tallinn",
"Europe/Tirane",
"Europe/Tiraspol",
"Europe/Uzhgorod",
"Europe/Vaduz",
"Europe/Vatican",
"Europe/Vienna",
"Europe/Vilnius",
"Europe/Warsaw",
"Europe/Zagreb",
"Europe/Zaporozhye",
"Europe/Zurich",
"Hongkong",
"Iceland",
"Indian/Antananarivo",
"Indian/Chagos",
```

```
"Indian/Christmas",
"Indian/Cocos",
"Indian/Comoro",
"Indian/Kerguelen",
"Indian/Mahe",
"Indian/Maldives",
"Indian/Mauritius",
"Indian/Mayotte",
"Indian/Reunion",
"Iran",
"Israel",
"Jamaica",
"Japan",
"Libya",
"Mexico/BajaNorte",
"Mexico/BajaSur",
"Mexico/General",
"Pacific/Apia",
"Pacific/Auckland",
"Pacific/Chatham",
"Pacific/Easter",
"Pacific/Efate",
"Pacific/Enderbury",
"Pacific/Fakaofo",
"Pacific/Fiji",
"Pacific/Funafuti",
"Pacific/Galapagos",
"Pacific/Gambier",
"Pacific/Guadalcanal",
"Pacific/Guam",
"Pacific/Honolulu",
"Pacific/Johnston",
"Pacific/Kiritimati",
"Pacific/Kosrae",
"Pacific/Kwajalein",
"Pacific/Majuro",
"Pacific/Marquesas",
"Pacific/Midway",
"Pacific/Nauru",
"Pacific/Niue",
"Pacific/Norfolk",
"Pacific/Noumea",
"Pacific/Pago Pago",
"Pacific/Palau",
"Pacific/Pitcairn",
"Pacific/Ponape",
"Pacific/Port Moresby",
"Pacific/Rarotonga",
```

```
"Pacific/Saipan",
"Pacific/Samoa",
"Pacific/Tahiti",
"Pacific/Tarawa",
"Pacific/Tongatapu",
"Pacific/Truk",
"Pacific/Wake",
"Pacific/Wallis",
"Pacific/Yap",
"Poland",
"Portugal",
"Singapore",
"Turkey",
"US/Alaska",
"US/Aleutian",
"US/Arizona",
"US/Central",
"US/East-Indiana",
"US/Eastern",
"US/Hawaii",
"US/Indiana-Starke",
"US/Michigan",
"US/Mountain",
"US/Pacific",
"US/Samoa",
"UTC"
```

Appendix 2

HTTP error codes

Following standard HTTP Error codes are returned by the REST APIs:

1. 200 – operation is successful.
2. 401 – unauthorized access for a resource – users must login first.
3. 404 – resource not found.
4. 400 – bad request – some of the information sent to the appliance is not valid.
5. 500 – internal server error – a software bug on the appliance