API Developer Guide

for the

VeloCloud Orchestrator

Version History

VCO Version(s)	3.3.x
Publication Date	4/11/2019
Contact	support@velocloud.net

Table of Contents

Who Should Read this Document

This guide provides an overview of the functionality of the VeloCloud Orchestrator Portal API, and the data model on which it operates. It is intended for consumption by network administrators (and delegates thereof) affiliated with VeloCloud's service provider partners and customers. It assumes some baseline familiarity with concepts and technologies related to web APIs, such as HTTP, cookies, JSON, etc.

This document is not intended to describe specific API methods in detail; readers should refer the Swagger API reference documentation for a list of available methods and their parameterizations.

Copyright

Trademarks

Software License Agreement

Disclaimer

Introduction

The VeloCloud Orchestrator (VCO) powers the management plane in the VeloCloud SD-WAN solution. It offers a broad range of configuration, monitoring and troubleshooting functionality to enterprises and service providers alike. The VCO Portal is the web service through which administrators manage all network- and device-level configuration and query network and device state. The Portal HTTP API exposes a suite of methods that may be invoked via either JSON-RPC or REST-like requests, as described in the following section of this document.

Architecture

The VCO API exclusively accepts HTTP POST calls via the `/portal` URL path (e.g. `https://my-orchestrator.velocloud.net/portal`). Consistent with the JSON-RPC specification (v2.0), the API enforces that request bodies must consist of a method name (`"method"`), a parameters object (`"params"`), a user-specified unique request identifier (`"id"`, by convention an integer such as a ms-precision epoch timestamp), and the version of the JSON-RPC specification to which the request adheres (`"jsonrpc"`). The VCO supports only the 2.0 iteration of the JSON-RPC specification, and so the value of the `jsonrpc` parameter should *always* be the string `"2.0"`. The request body should of course be JSON-encoded.

```
curl -H 'application/json' -d
'{"jsonrpc":"2.0","method":"event/getEnterpriseEvents","params":{"enterpriseId"
:1},"id":1}' --cookie cookies.txt -X POST https://vcoX.velocloud.net/portal/
```

A sample JSON-RPC request, constructed with curl

The VCO Portal also accepts a shorthand form of JSON-RPC calls via the `/portal/rest/` base path. This interface is meant to eliminate some of the protocol “overhead” required by the standard JSON-RPC interface, and may feel more familiar to those familiar with URL-based REST semantics. Clients are free to make requests in either format; there are no functional limitations specific to either one.

In processing REST-like requests, the VCO parses the JSON-RPC `method` name from the portion of the URL path appearing after `/portal/rest/`. It interprets the request body in precisely the same way that it does the JSON-RPC `params`.

```
curl -H 'application/json' -d '{"enterpriseId":1}' --cookie cookies.txt -X POST
https://vcoX.velocloud.net/portal/rest/event/getEnterpriseEvents
```

A sample shorthand-RPC request, constructed with curl

Authentication

As of the current release, the VCO API supports only cookie-based authentication. A client initiates a session by invoking either the `login/enterpriseLogin` or `login/operatorLogin` method, depending on the user class (Partner users should use the former method). Upon successful authentication, these methods produce a response with a Set-Cookie header, from which a `velocloud.session` cookie may be parsed. The VCO uses this token (which must be passed in a Cookie header) to authenticate subsequent requests from the client. Session cookies expire after a operator-configurable period of time (24 hours, by default) and may be refreshed by invoking the login method again.

General Usage

HTTP Response Codes

The VCO usage of HTTP response code aims to be consistent with this draft JSON-RPC over HTTP specification. Response codes are enumerated in the Swagger document.

Datetime Formats

The VCO accepts the following date formats:

- 13-digit millisecond-precision epoch timestamps (e.g. `1500000000000`)
- Datetime strings formatted consistently with RFC 3339.

Query Intervals

The VCO exposes time series data (e.g. device system health metrics such as CPU and memory usage, network metrics such as latency/jitter/loss, volumetric traffic flow data) via various API methods. By default, Edges and Gateways report new statistics every five minutes. Due to various factors (clock drift, network jitter, server-side processing delays), statistics associated with a given interval beginning at time X are often not reflected in API output until time X + 10 minutes. **As such, we do not recommend adopting query intervals smaller than 10 minutes in time**.

Client Setup & Usage

Demo: Postman

This section describes the basic steps required to access and use the VCO API using an HTTP client tool. This section uses the Postman HTTP client. However, you can interact with the VCO API using any HTTP client that supports cookies. The key points (authentication, enabling cookies, and making API calls) are also applicable to other HTTP clients.

Enabling Cookie Support

Older, browser-based versions of Postman did not enable native support for cookies by default. In order to enable support, you may need to install and enable Postman's Interceptor extension. To do so, you can simply click the satellite icon in the top navigation (the orange one in the image below) and follow the installation instructions.

Authentication

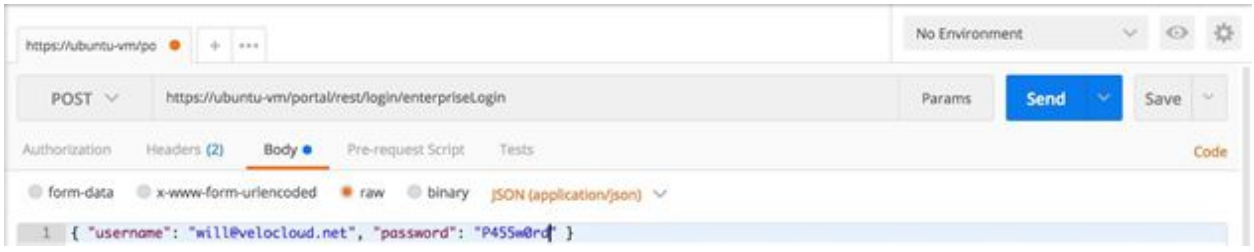
The VCO API uses HTTP cookies as a means of authenticating users of the portal service.

Login

Use the login method that matches your credential type:

- enterprise and partner (MSP) users: `login/enterpriseLogin`
- operator users: `login/operatorLogin`

No request headers are explicitly required.

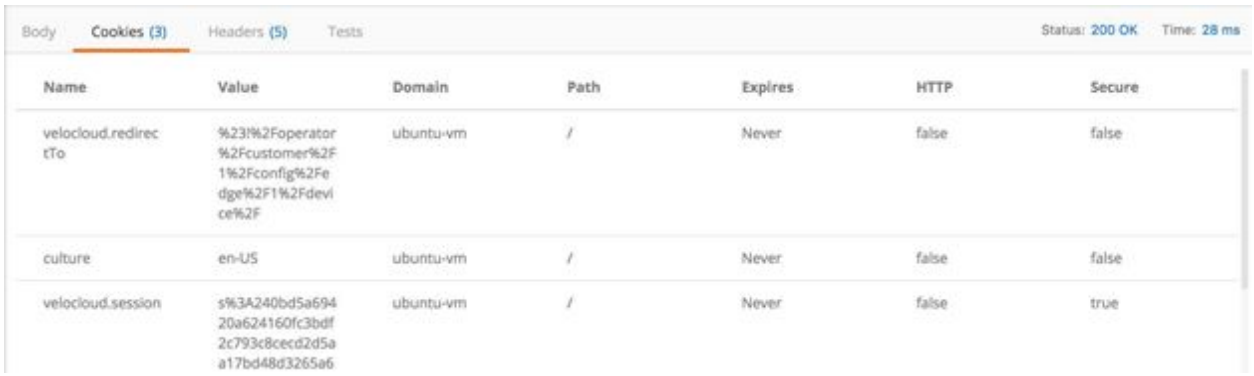


These methods return a pair of cookies in Set-Cookie HTTP response headers.

- `velocloud.message` is used to provide feedback when login credentials are invalid. A non-empty velocloud.message cookie indicates a login failure.
- `velocloud.session` is populated with a session cookie upon successful login.

Successful Login Response

A successful response returns an HTTP 200 header and a `velocloud.session` cookie. Example:



Body Cookies (3) Headers (5) Tests Status: 200 OK Time: 28 ms

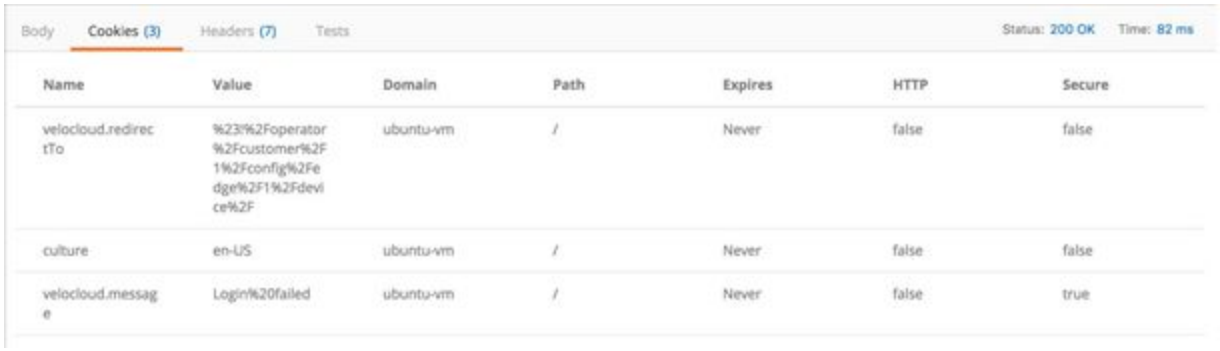
Name	Value	Domain	Path	Expires	HTTP	Secure
velocloud.redirectTo	%23!%2Foperator%2Fcustomer%2F1%2Fconfig%2Fedge%2F1%2Fdevice%2F	ubuntu-vm	/	Never	false	false
culture	en-US	ubuntu-vm	/	Never	false	false
velocloud.session	s%3A240bd5a69420a624160fc3bdf2c793c8cecd2d5aa17bd48d3265a6	ubuntu-vm	/	Never	false	true

```
Set-Cookie: velocloud.message=; Path=/; Expires=Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: velocloud.session=s%<LONG_BASE64_ENCODED_STRING>; Path=/; Secure
```

Your `velocloud.session` cookie must be passed in an HTTP Cookie header on subsequent API calls. Be warned that session cookies expire after 24 hours under the default VCO configuration. Note that Postman hides empty cookies, so the empty velocloud.message cookie is not displayed below.

Unsuccessful Login Response

An unsuccessful response returns these cookies as well. However, the `velocloud.message` cookie indicates that an authentication error has occurred. Example:



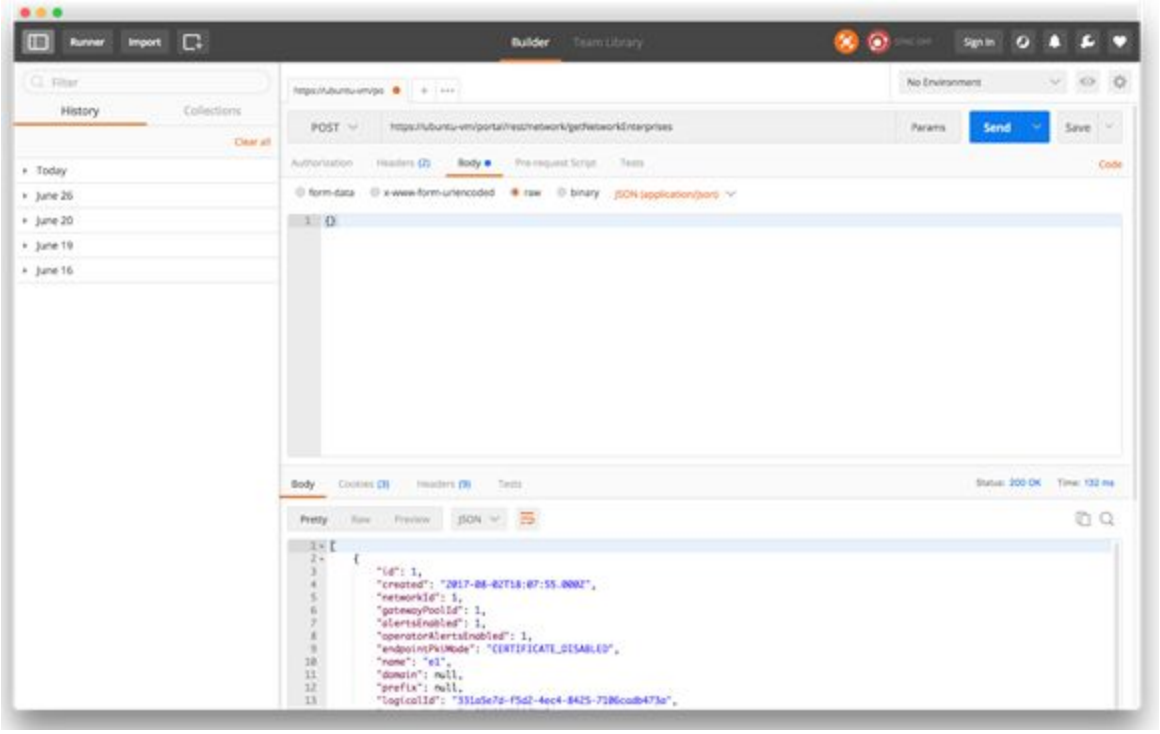
Body Cookies (3) Headers (7) Tests Status: 200 OK Time: 82 ms

Name	Value	Domain	Path	Expires	HTTP	Secure
velocloud.redirectTo	%23!%2Foperator%2Fcustomer%2F1%2Fconfig%2Fedge%2F1%2Fdevice%2F	ubuntu-vm	/	Never	false	false
culture	en-US	ubuntu-vm	/	Never	false	false
velocloud.message	Login%20failed	ubuntu-vm	/	Never	false	true

```
Set-Cookie: velocloud.message=Login%20failed; Path=/; Secure
Set-Cookie: velocloud.session=; Path=/; Expires=Thu, 01 Jan 1970 00:00:00 GMT
```

Making API Calls

After authenticating and setting your cookie header, you can simply make REST calls as you usually would.



Demo: curl

The rest of this section shows examples of invoking VCO API methods with the curl command-line client.

Authentication

The `--cookie-jar` option may be used to specify a location on your filesystem where curl can store cookies for the purpose of authenticating subsequent API calls. For demonstration we use `/tmp/cookies.txt`.

```
curl --cookie-jar /tmp/cookies.txt -k -d
'{"username":"test@test.com","password":"s3cret"}'
https://vcoX.velocloud.net/portal/rest/login/operatorLogin

curl --cookie-jar /tmp/cookies.txt -k -d
'{"username":"test@test.com","password":"s3cret"}'
https://vcoX.velocloud.net/portal/rest/login/enterpriseLogin
```

The same guidelines from the Postman demonstration in the previous section apply, with respec to how the results of these calls should be interpreted.

Making API Calls

Fetch All Edge Events

The following request fetches all edge events in the user's enterprise context.

```
curl -d '{}' --cookie /tmp/cookies.txt
https://vcoX.velocloud.net/portal/rest/event/getEnterpriseEvents

curl -d '{"jsonrpc":"2.0","method":"event/getEnterpriseEvents","params":{},"id":1}'
--cookie cookies.txt https://vcoX.velocloud.net/portal/
```

You can parameterize this call with an interval or specific edge id:

```
curl -d '{"interval":{"start":1428047513379},"edgeId":1}' --cookie cookies.txt
https://vcoX.velocloud.net/portal/rest/event/getEnterpriseEvents
curl -d
'{"jsonrpc":"2.0","method":"event/getEnterpriseEvents","params":{"interval":{"start":14
28047513379},"edgeId":1},"id":2}' --cookie cookies.txt
https://vcoX.velocloud.net/portal/
```

Sample Response


```
{
  "result": {
    "metaData": {
      "limit": 2048,
      "more": false
    },
    "data": [
      {
        "id": 2439,
        "eventTime": "2015-04-03T23:26:29.000Z",
        "event": "USER_LOGIN",
        "category": "USER",
        "severity": "INFO",
        "message": "test@test.com from [127.0.0.1]",
        "detail": null,
        "enterpriseUsername": "test@test.com",
        "edgeName": null
      },
      {
        "id": 1475,
        "eventTime": "2015-04-03T21:58:20.000Z",
        "event": "EDGE_PROVISION",
        "category": "EDGE",
        "severity": "INFO",
        "message": "activation key: 93WA-Y6UN-3ANT-CL2G",
        "detail": null,
```

```
        "enterpriseUsername": "test@test.net",
        "edgeName": "e2"
      },
      {
        "id": 1474,
        "eventTime": "2015-04-03T21:56:00.000Z",
        "event": "EDGE_PROVISION",
        "category": "EDGE",
        "severity": "INFO",
        "message": "activation key: Y5ED-XBEK-9ZFP-E4VE",
        "detail": null,
        "enterpriseUsername": "test@test.net",
        "edgeName": "e2"
      }
    ]
  },
  "id": 7
}
```


Retrieve Current Edge Link Status

The following request retrieves the current edge link status for all edge links in the user's enterprise context.

```
curl -d '{}' --cookie cookies.txt
https://vcoX.velocloud.net/portal/rest/monitoring/getEnterpriseEdgeLinkStatus

curl -d
'{"jsonrpc":"2.0","method":"monitoring/getEnterpriseEdgeLinkStatus","params":{}
,"id":1}' --cookie cookies.txt https://vcoX.velocloud.net/portal/
```

Sample Response


```
{
  "result": [
    {
      "enterpriseName": "Test Enterprise",
      "enterpriseId": 1000,
      "enterpriseProxyId": null,
      "enterpriseProxyName": null,
      "edgeName": "Branch-1-Edge",
      "edgeState": "CONNECTED",
      "edgeSystemUpSince": "2017-01-10T16:38:29.000Z",
      "edgeServiceUpSince": "2017-01-11T19:37:55.000Z",
      "edgeLastContact": "2017-01-19T23:50:39.000Z",
      "edgeId": 19271,
      "edgeSerialNumber": "VC05200002083",
      "edgeModelNumber": "edge520",
      "isp": "ISPCo",
      "interface": "GE1",
```

```
      "linkState": "STABLE",
      "linkLastActive": "2017-01-19T23:48:19.000Z",
      "linkVpnState": "STABLE",
      "linkId": 45908
    },
    {
      "enterpriseName": "Test Enterprise",
      "enterpriseId": 1000,
      "enterpriseProxyId": 32,
      "enterpriseProxyName": "Branded MSP",
      "edgeName": "A brand new edge",
      "edgeState": "NEVER_ACTIVATED",
      "edgeSystemUpSince": "2016-05-03T21:25:09.000Z",
      "edgeServiceUpSince": "0000-00-00 00:00:00",
      "edgeLastContact": "0000-00-00 00:00:00",
      "edgeId": 4123,
      "edgeSerialNumber": null,
      "edgeModelNumber": "edge500",
      "isp": null,
      "interface": null,
      "linkState": null,
      "linkLastActive": null,
      "linkVpnState": null,
      "linkId": null
    }
  ],
  "id": 1
}
```


Get All Gateways

The following request retrieves all gateways in the operator context.

```
curl -d '{"id": 4, "with": ["site","pools"]}' --cookie cookies.txt
https://vcoX.velocloud.net/portal/rest/network/getNetworkGateways

curl -d '{"jsonrpc":"2.0","method":"network/getNetworkGateways","params":{
"id": 4, "with": ["site","pools"] },"id":1}' --cookie cookies.txt
https://vcoX.velocloud.net/portal/
```

Note the following:

- You can optionally filter results by network (pass a `networkId`) or by gateway (pass an array of `gatewayIds`).
- You can optionally retrieve site and pool details by including `"site"` and `"pools"`, respectively, in the `"with"` clause, as shown in the above request.
- Other valid `"with"` options include `"enterprises"`, `"enterpriseAssociations"`, `"dataCenters"`, `"certificates"`, `"handOffEdges"`, and `"roles"`.

Sample Response


```
{
  "result": [
    {
      "id": 4,
      "created": "2015-06-17T00:05:30.000Z",
      "networkId": 1,
      "siteId": 4,
      "activationKey": "ZR4H-5W63-EZDZ-D8N7",
      "activationState": "ACTIVATED",
      "activationTime": "2015-06-17T01:08:29.000Z",
      "softwareVersion": "",
      "buildNumber": "",
      "utilization": 0,
      "utilizationDetail": {},
      "connectedEdges": 0,
      "deviceId": "5.5.5.5",
      "logicalId": "gateway.6a8f83d6-6b2e-4868-8e95-7abf2b369def",
      "name": "Oregon",
      "gatewayState": "CONNECTED",
      "alertsEnabled": 1,
      "description": null,
      "dnsName": "__keep_alive__",
      "isLoadBalanced": 0,
      "privateIpAddress": "",
      "ipAddress": "5.5.10.5",
      "lastContact": "2015-06-17T01:08:30.000Z",
      "systemUpSince": "2015-06-17T01:06:46.000Z",
      "serviceUpSince": "2015-06-17T01:08:26.000Z",
      "serviceState": "IN_SERVICE",
      "endpointPkiMode": "CERTIFICATE_DISABLED",
      "handOffDetail": null,
      "ipsecGatewayDetail": null,
      "modified": "2015-06-17T01:11:18.000Z",
      "site": {
        "id": 4,
        "created": "2015-06-17T00:05:30.000Z",
        "name": null,
        "contactName": "VeloAcme Operator",
        "contactPhone": null,
        "contactMobile": null,
        "contactEmail": "operator@veloacme.net",
        "streetAddress": null,
        "streetAddress2": null,
        "city": "Boardman",
        "state": "OR",
        "postalCode": "97818",
        "country": "US",
        "lat": 45.839901,
```

```
        "lon": -119.7006,
        "timezone": "America/Los_Angeles",
        "locale": "en-US",
        "shippingSameAsLocation": 1,
        "shippingContactName": null,
        "shippingAddress": null,
        "shippingAddress2": null,
        "shippingCity": null,
        "shippingState": null,
        "shippingPostalCode": null,
        "shippingCountry": null,
        "modified": "2015-06-17T00:05:30.000Z"
      },
      "pools": [
        {
          "gatewayPoolAssocId": 9,
          "gatewayId": 4,
          "id": 1,
          "networkId": 1,
          "created": "2015-06-16T23:57:17.000Z",
          "name": "Default Pool",
          "description": "XX gateway pool used when none is explicitly assigned to an
enterprise",
          "isDefault": 1,
          "handOffType": "ALLOW",
          "modified": "2016-11-03T11:12:38.000Z"
        },
        {
          "gatewayPoolAssocId": 285,
          "gatewayId": 4,
          "id": 176,
          "networkId": 1,
          "created": "2016-08-01T23:08:20.000Z",
          "name": "A new test pool",
          "description": "",
          "isDefault": 0,
          "handOffType": "ALLOW",
          "modified": "2016-08-01T23:52:35.000Z"
        }
      ]
    },
  ],
  "id": 1
}
```


VCO API Functionality

This section provides a functional overview of the VCO API as well as information to help you interact with the API more effectively.

Profile Configuration

Client applications can use the VCO API to configure and manage enterprise and device profiles.

About Enterprise Profiles

Before you begin to configure and manage Edge policies via the VCO API, you should understand VeloCloud *enterprise profiles*, which are defined by a simple JSON object that contains:

- a name and description
- four associated *modules*
- a set of related object references (*refs*)

Configuration Profile JSON

```
{ "id": 5,
  "created": "2017-04-04T21:43:11.000Z",
  "name": "Quick Start Internet",
  "version": "1491342191080",
  "description": "Out of the box direct to internet profile",
  "effective": "2016-02-14T21:41:39.000Z",
  "modified": "2016-02-14T21:43:11.000Z",
  "modules": [ … ],
  "refs": [ … ]
}
```

In the VCO Web UI, enterprise profiles are displayed on the **Configuration** → **Profiles** page. Use the `enterprise/getEnterpriseConfigurations` API to retrieve them. You can optionally request `modules` and `refs` by specifying `{"with": ["modules", "refs"]}` in the request parameters.

Enterprise Profile Modules

An enterprise profile has four associated modules.

Module	Description
`deviceSettings`	• Cloud VPN, wired and wireless interfaces, addressing and services configuration • Accessed on the VCO user interface under the **Device** tab
`QOS`	• SD-WAN business policy rules • Configured on the VCO user interface under the **Business Policy** tab

`firewall`	• Inbound, outbound and service access rules • Configured on the VCO user interface under the **Firewall** tab
`WAN`	• Reserved for future functionality. Currently unused at the profile level.

Call the `edge/edgeProvision` method to provision an Edge. The required `configurationId` designates an existing enterprise profile to be assigned to the Edge.

As part of provisioning, the VCO creates two distinct configurations:

- a relational linkage between the selected profile and the new Edge
- a new *Edge-specific profile* that it uses to store the Edge-level configuration and overrides to the enterprise profile

Together, these two configurations (and the modules containing the actual configuration data) comprise the Edge's configuration "stack".

Call the `edge/getEdgeConfigurationStack` method to retrieve the complete stack. The resulting two-entry array, composed of `[<Edge Specific Profile>, <Enterprise Profile>]`, in that order, contains all of the data that is rendered on the VCO's **Configuration** → **Edge** page. Meanwhile, when the Edge receives configuration updates from the VCO, it receives a merged composite of its assigned enterprise profile and the Edge-specific configuration.

Edge-specific Profile Modules

Edge-specific profiles may contain up to five associated modules.

Module	Description
`deviceSettings`	• Edge addressing and interface configuration, high-availability and static routes • Always present at the Edge level
`WAN`	• WAN overlay and link configuration • Always present at the Edge level
`controlPlane`	• Cloud VPN and Edge routing • **Important**: *Do not change this module using the VCO API*. The `controlPlane` module is auto-generated by the VCO and dynamically recalculated as routing changes are reported to the VCO (for example, for OSPF or BGP learned routes). • Following activation, always present at the Edge level
`firewall`	• Rules are pre-pended to the ruleset inherited from the enterprise profile

	• Present only when Edge-level firewall rules are defined
QOS	• Rules are pre-pended to the ruleset inherited from the enterprise profile • Present only when Edge-level QOS rules are defined

Network Segment-aware Profiles

VCO release 3.0 introduced network segment-aware profiles. When an enterprise is created, the creator (a network Operator or Partner administrator) designates either of the following:

- a **network allocation-based** profile (legacy), or
- a **network segment-aware** operator profile for the enterprise

A new enterprise-level profile is generated accordingly. The profile can in turn be assigned to one or more Edges.

Note: An enterprise can support only one type of profile. It cannot simultaneously support both allocation-based and segment-aware profiles because the configuration modules structure varies between them.

Configuration Module Structure

Configuration modules share many common properties.

Configuration Module JSON


```
{
  "configurationId": 12,
  "created": "2017-08-02T20:35:49.000Z",
  "data": { ... },
  "description": null,
  "draftComment": null,
  "draftCreated": "0000-00-00 00:00:00",
  "draftData": null,
  "effective": "0000-00-00 00:00:00",
  "id": 69,
  "isSmallData": 1,
  "modified": "2017-08-02T20:35:49.000Z",
  "name": "deviceSettings",
  "previousCreated": "0000-00-00 00:00:00",
  "previousData": null,
  "schemaVersion": "2.0.0",
  "type": "ENTERPRISE",
  "version": "1501706149569"
}
```

Configuration Data

From the perspective of API clients, all module attributes are read-only except `data`, which contains the actual configuration data. The JSON structure of the module `data` object can vary widely, according to the following factors:

- Whether the module belongs to an enterprise profile or Edge-level profile. In either case, the module `type` will be `ENTERPRISE`. The most reliable way to distinguish between enterprise and Edge-specific configuration modules is to check the name of the configuration to which they belong by calling `configuration/getConfiguration`.
- Whether certain features are enabled or not. Some JSON attributes are optional. If not specified (omitted), the feature is assumed to be disabled.
- Whether the enterprise supports segment-aware profiles.

To configure Edge behavior using the VCO APIs, you need to understand how enterprise profile and Edge-level modules vary in structure. The following sections describe these structures in detail. Module schemas are abbreviated for brevity. For complete schema specifications, refer to the Swagger documentation at code.vmware.com.

Firewall Modules

The firewall module always appears in enterprise profiles and appears at the Edge level only when Edge-specific overrides (rules or services) are configured.

Firewall Modules in Allocation-based Profiles

Firewall Module `data` JSON

```
{
  "firewall_enabled": true,
  "firewall_logging_enabled": false,
  "inbound": [ .. ],
  "outbound": [ .. ],
  "services": {
    "loggingEnabled": false,
    "ssh": {..},
    "localUi": {..},
    "snmp": {..},
    "icmp": {..}
  }
}
```

The firewall module shares many of the same properties in the enterprise profile and the Edge-specific profile, with the following considerations:

Property	Configuration	
	Enterprise	Edge
`firewall_enabled`	Required	Optional (overrides enterprise profile)
`firewall_logging_enabled`	Required	Optional (overrides enterprise profile)
`inbound` Inbound firewall rules are defined only at the Edge level because port forwarding and one-to-one NAT rules require Edge-specific address parameters.	N/A	Optional
`outbound`	Required	Optional (prepended to profile rules)
`services` When firewall services are configured at the Edge level, the Edge-level configuration completely overrides the enterprise-level configuration. Values must be specified at the Edge level for **all** of the properties that appear in the enterprise configuration.	Required	Optional (overrides enterprise profile)

Firewall Modules in Segment-Aware Profiles

Segment-aware profiles, and the Edges to which they are applied, have a slightly different firewall module.

Segment-Aware Firewall Module `data` JSON

```
{
  "firewall_enabled": true,
  "inbound": [ .. ],
  "inboundLoggingEnabled": false,
  "segments": [
    {
      "firewall_logging_enabled": false,
      "outbound": [ .. ],
      "segment": {
        "name": "Global Segment",
        "segmentId": 0,
        "segmentLogicalId": "<UUID>",
        "type": "REGULAR"
      }
    }
  ],
  "services": {
    "loggingEnabled": false,
    "ssh": {..},
    "localUi": {..},
    "snmp": {..},
    "icmp": {..}
```

```
  }
}
```

In a segment-aware firewall module, segment-specific configuration details are included in an `segments` array (which is always required). In the associated configuration data, outbound firewall rules and logging are configured on a per-segment basis. Inbound rules are also applied to per-segment sources via a `segmentId` property in a rule's `action` specification (not shown in the above code example).

QOS Modules

The QOS module consists of business policy rules, Class of Service (CoS) settings, and SD-WAN rate-limiting settings. It shares many common properties in enterprise profiles and Edge-specific profiles. The QOS module always appears in enterprise profiles and appears at the Edge level only when Edge-specific overrides are configured.

QOS Modules in Allocation-based Profiles

Allocation-based profiles use a QOS module structure that differs slightly from segment-based profiles. The following sample QOS module is an example of an allocation-based profile.

QOS Module `data` JSON

```
{ "rules": [{
    "name": "Box",
    "match": { "classid": -1, "sip": "any", ... },
    "action": {
      "edge2CloudRouteAction": {..},
      "edge2DataCenterRouteAction": {..},
      "QoS": {..},
      "edge2EdgeRouteAction": {..},
      "sla": {..},
      "nat": {..},
      "routeType": "edge2any",
    }
  }, ... ],
  "defaults": [{..}],
  "cosMapping": {
    "lsInputType": "weight",
    "realtime": {..},
    "transactional": {..},
    "bulk": {..}
  },
  "serviceRateLimit": {
    "enabled": false,
    "inputType": "percent",
    "value": 1
  },
  "webProxy": {..}
}
```

The following table describes the QOS module differences between enterprise profile-level modules and Edge-level profiles.

Property	Configuration	
	Enterprise	Edge
`rules` QoS `rules` consist of: • a `name` • a `match` object identifying the traffic to which the rule applies (such as application category, source/destination IP, or protocol) • an `action` prescribing how the traffic should be handled.	Optional	Optional (appended to enterprise profile rules)
`defaults` Operator-defined QoS rules that adhere to the same schema as the `rules` defined by enterprise users. `defaults` do not appear in the Edge-level QoS module.	Required	N/A
`cosMapping` These settings determine the traffic service class-to-weight mappings prescribed by the “Low”, “Normal”, and “High” labels that are used to prioritize traffic in QoS rules. Throttling can also be configured.	Required	Optional (overrides enterprise profile)
`serviceRateLimit` Allows you to designate a enterprise profile-level or Edge-specific rate limit for traffic traversing the SD-WAN overlay network.	Optional	Optional (overrides enterprise profile)
`webProxy` Currently unused.	N/A	N/A

QOS Modules and Segment-aware Profiles

Segment-aware profiles, and the Edges to which they are applied, adopt a slightly different QOS module structure. The `cosMapping` and `rules`/`defaults` are configured on a per-segment basis.

Segment-Aware QOS Module `data` JSON

```
{ "segments": [{
    "rules": [..],
    "defaults": [..],
    "cosMapping": {..},
    "webProxy": {..},
    "segment": {
```

```
      "name": "Global Segment",
      "segmentId": 0,
      "segmentLogicalId": "<UUID>",
      "type": "REGULAR"
    }
  }, ... ],
  "serviceRateLimit": {
    "enabled": false,
    "inputType": "percent",
    "value": 1
  }
}
```

DeviceSettings Module

The `deviceSettings` module configures a variety of functions, including interface addressing, routing, high availability, WiFi radio, NTP, SNMP, DNS, VQM, netflow, VPN, BGP and OSPF. It is always present in both enterprise profiles and Edge-specific profiles, although the structure of the module varies significantly depending on where it appears in the stack. Abbreviated examples are included below. For subsection schema details, please refer to the Swagger API reference documentation on code.vmware.com.

Enterprise Profile Device Settings Module `data` JSON in Allocation-based Profiles

```
{ "lan": {..},              // addressing schema and assignable VLANS
  "vpn": {..},              // cloud VPN enable and configuration (optional)
  "ospf": {..},             // OSPF enable and configuration
  "bgp": {..},              // BGP enable and configuration
  "dns": {..},              // DNS public and private providers
  "snmp": {..},             // SNMP enable and configuration
  "authentication": {..},   // 802.1x authentication providers
  "softwareUpdate": {..},   // currently unused (optional)
  "radioSettings": {..},    // WIFI radio country and power settings
  "netflow": {..},          // netflow enable and collector configuration (optional)
  "vqm": {},                // voice quality monitoring enable and collector (optional)
  "multiSourceQos": {},
  "models": {               // per model routed and LAN interface configuration
    "edge500": {
      "routedInterfaces": [],
      "lan": {
        "interfaces": []
      }
    },
    ... additional models ...
  }
}
```

Edge Device Settings Module `data` JSON in Allocation-based Profiles

```
{ "lan": {
    "networks": []           // required LAN addressing and VLAN configuration
    "interfaces": []         // optional, only present when interfaces are overridden
```

```
  },
  "routedInterfaces": [],     // required
  "routes": {                 // routes section is optional, as are all subsections
    "icmpProbes": [],
    "icmpResponders": [],
    "static": []              // static routes
  },
  "ha": {},                   // optional, HA is enabled at edge level only
  "multiSourceQos": {},       // optional, present only as edge override
  "radioSettings": {},        // optional, present only as edge override
  "dns": { },                 // optional, present only as edge override
  "authentication": {},       // optional, present only as edge override
  "snmp": {},                 // optional, present only as edge override
  "netflow": {}               // optional, present only as edge override
}
```

Properties Comparison

The following table compares the enterprise-level and Edge-level properties in the `deviceSettings` module schema.

Property	Configuration	
	Enterprise	Edge
`authentication` Configured via a reference (or `ref`) to an externally-defined network service.	Required	Optional (overrides enterprise profile)
`bgp` Supported only when the BGP capability has been enabled on an enterprise.	Optional	Optional (overrides enterprise profile)
`dns` Configured via a set of references (or `ref`s) to externally-defined DNS services.	Required	Optional (overrides enterprise profile)
`ha`	N/A	Optional
`lan` By default, Edge LAN interface configurations are inherited from the model-specific `models` section of the profile. The presence of the `interfaces` property at the Edge level indicates that the profile-level configurations are overridden. Meanwhile, LAN `networks` are required at the Edge level, where per-edge addressing and VLANs are configured.	Required	Required
`models` Contains per-model LAN and routed (WAN) interface configurations. At the Edge level, the configuration items	Required	N/A

resolve to the `lan interfaces` configuration and `routedInterfaces` that are model specific.		
`multicast`	Optional	Optional (overrides enterprise profile)
`multiSourceQos` Requires that VPN be enabled.	Optional	Optional (overrides enterprise profile)
`netflow`	Optional	Optional (overrides enterprise profile)
`ntp`	Optional	Optional (overrides enterprise profile)
`ospf` Supported only when the OSPF capability has been enabled on an enterprise.	Optional	Optional
`radioSettings` Only 500-series Edges support WiFi radio, so profile-level `radioSettings` are applied only to those Edge models.	Required	Optional
`routedInterfaces`	Required	Required
`routes` Comprised of `static` routes, `icmpProbes`, and `icmpResponders`.	N/A	Optional
`snmp`	Optional	Optional (overrides enterprise profile)
`softwareUpdate` Currently unused.	Optional	N/A
`vpn` Cloud VPN configuration, which consists of Edge to non-VeloCloud site, Edge to Hub, and Edge to Edge.	Optional (N/A for Internet profiles)	N/A
`vqm` Supported only when the VQM capability has been enabled on an enterprise.	Optional (`ha` must be disabled)	Optional (overrides enterprise profile)
`vrrp` **New in VCO Release 3.1** To enable VRRP on an Edge, `ha` must be disabled.	N/A	Optional

Segment-aware Configurations

For segment-aware configurations, in the `deviceSettings` module, many settings (such as

BGP, DNS, and so on) are specified on a per-segment basis.

Segment-Aware Enterprise Profile Device Settings Module `data` JSON

```
{
  "lan": {..},
  "models": {
  "edge500": {
  "lan": {..},
  "routedInterfaces": [..]
},
"radioSettings": {..},
"segments": [{
  "authentication": {..},
  "bgp": {..},
  "dns": {..},
  "multiSourceQos": {..},
  "netflow": {..},
  "ntp": {..},
  "ospf": {..},
  "segment": {
    "name": "Global Segment",
    "segmentId": 0,
    "segmentLogicalId": "<UUID>",
    "type": "REGULAR"
  },
  "snmp": {..},
  "vpn": {..},
  "vqm": {..}
  }],
  "softwareUpdate": {..}
}
```

Segment-Aware Edge Device Settings Module `data` JSON

```
{ "lan": {
  "management": {..},
  "networks": {..}
},
"ha": {..},
"routedInterfaces": {..},
"segments": [{
  "authentication": {..},
  "bgp": {..},
  "dns": {..},
  "multiSourceQos": {..},
  "netflow": {..},
  "ntp": {..},
  "ospf": {..},
  "routes": {..},
  "segment": {
    "name": "Global Segment",
    "segmentId": 0,
    "segmentLogicalId": "<UUID>",
```

```
    "type": "REGULAR"
  },
  "snmp": {..},
  "vpn": {..},
  "vqm": {..}
  }],
  "softwareUpdate": {..}
}
```

WAN Modules

The WAN configuration module is defined only in Edge-specific profiles. It contains WAN overlay configuration details or, if no overlay is defined, link configuration details (such as static address assignment). The JSON consists of two array properties:

- `links` carries the active configuration
- `networks` is a legacy attribute used by older (pre 2.x) VeloCloud Edges

When configuring overlay or link configuration, only the `links` array needs to be updated. The VCO will automatically compute the `networks` if necessary.

```
{ "links": [
    {
      "logicalId": "00:00:34:01:11:33",
      "internalId": "1af4a5b3-e164-42ae-bdcb-880ecbeab484",
      "discovery": "USER_DEFINED",
      "mode": "PRIVATE",
      "type": "WIRED",
      "name": "MPLS PROVIDER",
      "isp": "AT&T",
      "publicIpAddress": null,
      "sourceIpAddress": null,
      "nextHopIpAddress": null,
      "customVlanId": false,
      "vlanId": 0,
      "virtualIpAddress": null,
      "dynamicBwAdjustmentEnabled": false,
      "bwMeasurement": "SLOW_START",
      "upstreamMbps": null,
      "downstreamMbps": null,
      "backupOnly": false,
      "udpHolePunching": false,
      "overheadBytes": 0,
      "MTU": 1500,
      "mplsNetwork": "",
      "dscpTag": "",
      "staticSlaEnabled": false,
      "classesofServiceEnabled": false,
      "encryptOverlay": true,
      "staticSLA": {
        "latencyMs": 0,
        "jitterMs": 0,
        "lossPct": 0
      },
```

```
      "classesOfService": {
        "classId": null,
        "classesOfService": []
      },
      "interfaces": [
        "INTERNET3"
      ],
      "lastActive": null
    }
  ],
  "networks": [
    {
      "logicalId": "00:00:34:01:11:33",
      "internalId": "1af4a5b3-e164-42ae-bdcb-880ecbeab484",
      "discovery": "USER_DEFINED",
      "mode": "PRIVATE",
      "type": "WIRED",
      "name": "MPLS PROVIDER",
      "isp": "AT&T"
    }
  ],
}
```

References (or "*refs*")

References (*refs*) are associations between network services (e.g. DNS providers, authentication services, VPN hubs) and profiles. Services may be used across many profiles, so they are defined as external, enterprise-global entities. *Refs* include objects that can be shared across multiple profiles (such as Networks or Network Services like DNS). These objects can include:

Feature	**Object(s)**
Network (overlapping or non-overlapping address scheme)	`deviceSettings:lan:allocation`

DNS services	`deviceSettings:dns:primaryProvider`
	`deviceSettings:dns:secondaryProvider`
	`deviceSettings:dns:privateProviders`

Authentication services	`deviceSettings:authentication`
Cloud Proxies	`deviceSettings:webProxy:provider`
Non-VeloCloud sites	`deviceSettings:vpn:dataCenter`
Edge Hubs	`deviceSettings:vpn:edgeHub` `deviceSettings:backHaulEdge`

Network Segments	`deviceSettings:segment`

Requesting References

When fetching a profile and its constituent modules using API methods such as `enterprise/getEnterpriseConfigurations` or `configuration/getConfiguration`, a client requests *refs* optionally by passing a `with` request parameter (as in `{..., "with": ["modules","refs"]}`). Each of the modules in the resulting configuration will contain a `refs` object, as shown in the example below. Each entry in the `refs` object is a set of typed associations between the module on which the object appears and a service (e.g. a DNS service, in the below example). These objects contain the *ref* `data` and a number of other related attributes for convenience. The structure and content of the `data` blob varies depending on the `refs` object and type.

Example Configuration Module *Refs*

```
"refs": {
  ...,
  "deviceSettings:dns:primaryProvider": {
    "id": 6,
    "enterpriseObjectId": 11,
    "configurationId": 5,
    "moduleId": 25,
    "ref": "deviceSettings:dns:primaryProvider",
    "data": {
      "primary": "8.8.8.8",
      "secondary": "8.8.4.4"
    },
    "modified": "2017-04-04T21:43:11.000Z",
    "version": "0",
    "object": "NETWORK_SERVICE",
    "name": "Google",
    "type": "dns",
    "logicalId": "d68acdeb-4b43-4e54-a00f-af64bbc4447b"
  }
}
```

Example: Update a Configuration Module (QoS)

Suppose you wanted to add a new QoS rule at the enterprise level to set the priority for RADIUS authentication traffic to “High”. The easiest way to manage this would be to copy a comparable rule and replace the identifier for the application class with the RADIUS application class identifier (which you can retrieve from the application map that can be downloaded from the Application Maps page). The following code examples assume that one such template rule is already defined for Box application traffic.

Disaster Recovery

Client applications can use the disaster recovery (DR) methods in the VCO API to configure the VCO for DR replication. DR management requires designating one VCO as Active and a secondary instance as a Standby. The disaster recovery configuration flow should follow roughly the same procedure that the VCO web GUI supports.

Edge

Client applications can use the VCO API to manage Edges (provisioning, activation, access, and more).

Enterprise

Client applications can use the enterprise methods in the VCO API to manage enterprises and related enterprise-level objects, including alert configurations, enterprise services, capabilities (such as BGP, OSPF, and PKI), and network allocations. To manage enterprise proxy users, see User Maintenance below.

Enterprise Proxy

Client applications can use the VCO API to manage Managed Service Provider (MSP) partner (enterprise proxy) settings. To manage enterprise proxy users, see User Maintenance below.

Event

Client applications can use the VCO API to retrieve operator or enterprise events in a given timeframe.

Firewall

Client applications can use the VCO API to retrieve the firewall logs for an enterprise.

Gateway

Client applications can use the VCO API to provision, delete, or update attributes on a gateway.

Link Quality Events

Client applications can use the VCO API to get link quality of experience (QoE) scores for a particular edge within some time interval. In the VCO GUI, link quality data is displayed on the **Monitor** → **Edges** page under the QoE tab.

Supported Link Quality Event API

Method	Description
`/linkQualityEvent/getLinkQualityEvents`	Returns link quality scores per link for a particular edge within a time interval. Rolls up link quality events to provide an aggregate score for the edge. Returns an empty array if no link quality events are available in the given timeframe.

Link Quality Event Response

This section provides some context on the returned results.

UUID Keys

At the top level, the response breaks down the data by link. The UUID keys you see (e.g. "`00000001-f7d9-48c2-9b56-ac549342be9b`") are link IDs. These IDs are static and you can get additional detail on the links to which they correspond by calling, for example, `edge/getEdge` specifying `{ "with": ["links"], ... }`.

Objects appearing in the response with keys 0,1,2 break out the data by traffic type, where **0 is voice, 1 is video, 2 is transactional**.

Time Series Metric Data

With respect to time series metric data, the following legend should be used to interpret the various `action`, `metric`, and state (`beforeState`, `afterState`) values:

Time Series Metric	Values
Action	0: NONE 1: AVOID 2: JITTER_BUFFER 3: ERROR_CORRECTION 4: FORWARD_ERROR_CORRECTION
Metric	0: LATENCY_RX

	1: LATENCY_TX 2: JITTER_RX 3: JITTER_TX 4: LOSS_RX 5: LOSS_TX
State	0: OFFLINE 1: UNKNOWN 2: RED 3: YELLOW 4: GREEN

Login

Client applications use the VCO API to login and authenticate.

Meta

Client applications can use the VCO API to retrieve metadata about any VCO API call.

Metrics

Client applications can use the VCO API to retrieve historical flow metrics, grouped by link, edge, application category, traffic destination, source OS, and so on.

Note: These methods are not intended for use in real-time monitoring applications.

Monitoring

Client applications can use the VCO API to monitor network state, including edge link utilization, events across enterprises, and BGP peer state.

Network

Client applications can use the VCO API to manage network and operator-level objects, including gateways, gateway pools, enterprises, and operator users.

Roles

Client applications can use the VCO API to list, create, or delete VCO user roles. Custom roles are created as a composition of privileges.

System Properties

Client applications can use the VCO API to manage system-wide VCO properties. These are visible using the Web interface on the **System Properties** page, which is accessible from the VCO Operator navigation menu.

User Maintenance

Client applications can use the VCO API to create, get, update, or delete a user. There are three types of users:

- operator users
- enterpriseProxy users (i.e. partner)
- enterprise (i.e. customer) users