# Versa Director REST API Overview

*For supported software information, click here.*

Versa Director provides REST APIs so that you can integrate upstream (northbound) applications. A number of integration options are available, allowing you to customize the integration of applications based on customer requirements.

The Director REST API includes the following features:

- APIs to automate and configure the Director node and its components and features
- REST APIs protected with username and password
- REST APIs protected with an open authorization (OAuth) token
- Options to view REST API syntax using the API browser, CLI, and web GUI

You can use any REST client to call the REST APIs. You use the **curl** command to run the REST APIs.

## Versa Networks Terminology

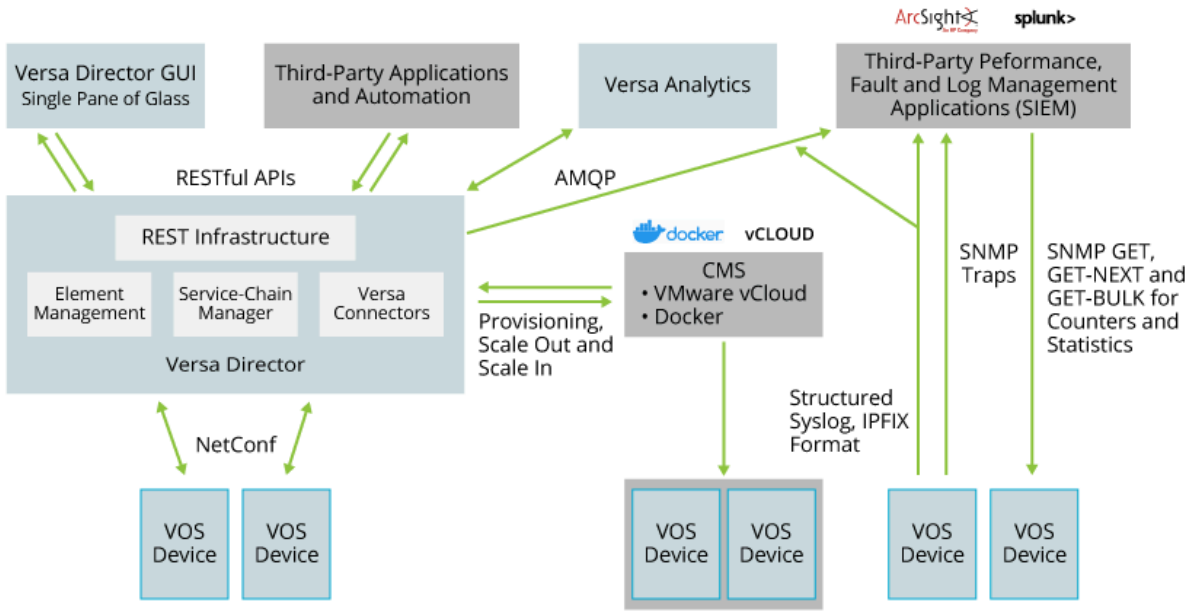
The following table defines the Versa Networks–specific terminology that is used in the API documentation.

| Versa Networks Term | ETSI or Industry Term | Description |
|---|---|---|
| Appliance or Versa Operating System™ (VOS™) device | VNF | Device running VOS software that can implement one or more virtual functions. |
| Bare-metal appliance or bare-metal device | VNF | Device either running directly on a commercial off-the-shelf (COTS) device or as a VM on which VOS software is preinstalled and which is discovered by a Director node. |
| Bronze model | Multitenant VNF | One or more NMS organizations associated with an instantiated VNF that implements vCPEs and network |

| Versa Networks Term | ETSI or Industry Term | Description |
|---|---|---|
| | | functions for multiple organizations in a VNF. |
| CMS connector | CMS | Cloud management systems such as vCloud Director that are used to implement virtualization. |
| CMS organization | Resource pool | CMS connector that has dedicated resources to implement network functions. |
| Director node | VNF manager | Management entity that communicates with both the orchestration layer and the VOS devices. |
| NMS organization | Customer | A network management system organization is associated with one or more CMS organization (resource pool). |
| Organization | Tenant | Customer entity for which a VNF is used to implement network functions. |
| Subscription plan | Per-tenant plan | Set of predefined plans that define a customer's network functions, elasticity policies, and other options. |
| Virtual appliance | VNF | Device that is instantiated by a Director node in the virtual data center using a VOS application template (OVF). |

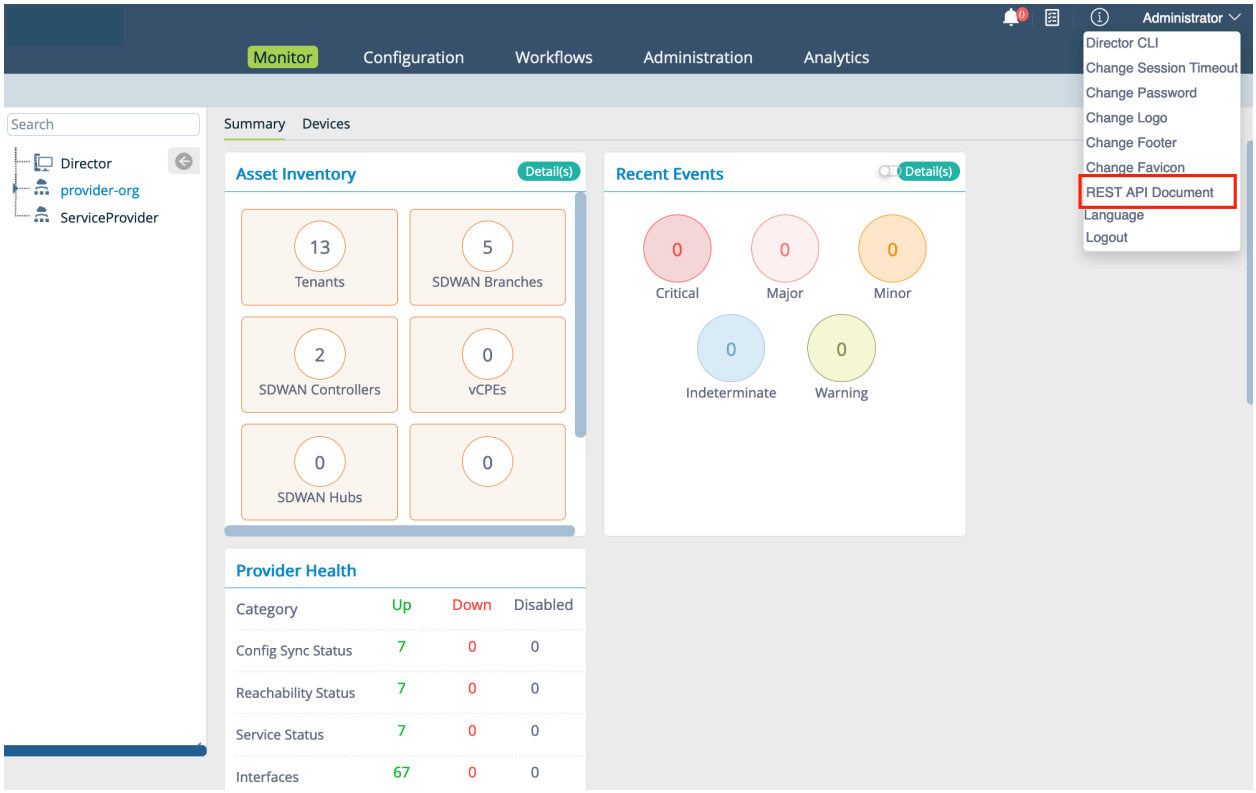
## Director Interfaces

You use a Director node to orchestrate and create VNFs, to set up an SD-WAN environment in the cloud and on bare-metal devices, and to manage the devices. Orchestrators and other upstream applications can interact with the Director node and the VNFs using the REST APIs. The following figure illustrates these functions.




## REST API URLs and Documentation

To access the Director REST APIs and API documentation from a Director node, click REST API Document in the user drop-down list in the top menu bar.




You can also download the Director REST APIs from the URL [https://upload.versa-networks.com/in...VFzV8ccqMzPEXN](https://upload.versa-networks.com/in...VFzV8ccqMzPEXN)

## REST API Categories

You can integrate an orchestrator with a Director node using one of the following models

- End customer must use the Director GUI.
- Orchestrator provides the GUI for Versa-specific operations.

## REST API Classes

A Director node provides REST APIs for the classes, or areas, described in the following table.

| REST API Category | Function Provide by API Calls |
|---|---|
| Connector management | • AQMP connector<br>• Authentication connector<br>• CMS cloud management |

| REST API Category | Function Provide by API Calls |
| --- | --- |
| | • Local CMS management<br>• Syslog connector |
| Director security | • API authentication using HTTPS basic authentication<br>• Secure API with OAuth 2.0 and OAuth token management |
| Director system administration | • HA configuration<br>• Rebooting<br>• Status checking<br>• User and role management |
| Director task status and information | |
| Fault management | • Alarms<br>• AMQP events |
| SD-WAN management | • Create a template<br>• Onboard a Controller node<br>• Onboard a device<br>• Onboard an organization |
| Tenant-specific service configuration | • ADC<br>• CGNAT<br>• IPsec (VPN)<br>• NGFW<br>• SD-WAN<br>• Stateful firewall |
| VOS device configuration management | • Interface management |

| REST API Category | Function Provide by API Calls |
|---|---|
| | • Organization management<br>• Predefined and security package metadata<br>• Routing<br>• Service node groups and elasticity policy |
| VOS device connection troubleshooting | • Check sync<br>• Ping<br>• Service status |
| VOS software package management | • List software packages<br>• Upgrade software packages<br>• Upload software packages |
| VOS device status monitoring | |

## REST API URL Conventions

The following table describes the URL conventions for the Director REST APIs. Director APIs support Get, Post, Patch, Put, and Delete HTTP operations for most configurations.

| API Type | URL | Name Server Control (NSC) Command Type |
|---|---|---|
| Configuration | GET https://127.0.0.1:9182/api/config | Show, in configuration mode |
| Configuration requests | POST https://127.0.0.1:9182/api/config | Request, in configuration or operational mode |
| Operational | GET https://127.0.0.1:9182/api/operational | Show, in operational mode |
| OAuth | GET https://127.0.0.1:9183/auth/ | Token-based APIs |

| API Type | URL | Name Server Control (NSC) Command Type |
|---|---|---|
| VNMS | GET https://127.0.0.1:9182/vnms/ | Custom APIs that interact with both the common database (CDB) and the Postgres database |

## APIs Derived from YANG Data Model

The path for APIs derived from YANG data model starts with either /api/config or /api/operational.

These APIs generate the following status and error codes:

| Status Code | Description |
|---|---|
| 200 | OK (success) |
| 201 | Created |
| 202 | Accepted |
| 204 | No content |
| 400 | Bad request |
| 401 | Unauthorized |
| 405 | Method not allowed |
| 415 | Unsupported media type |
| 500 | Internal error |

The following examples show calls to YANG-derived APIs and the values returned.

- Success code

```
curl -i -k -X GET 'https://director-ip-address:9182/api/config/nms/provider/organizations/
organization' -H "Accept: application/json" -H "Content-Type: application/json" -u Administrator:Versa@123
HTTP/1.1 200 OK
```

- Error code

```
curl -i -k -X POST 'https://director-ip-address:9182/api/config/system/syslog-servers' -H
"Accept: application/json" -H "Content-Type: application/json" -u Administrator:Versa@123 -d
'{"server":{"host":"ip","port":"WrongPort","enabled":true}}'
HTTP/1.1 400 Bad Request
{ "errors":
```

```
    { "error": [
        { "error-message": "invalid value for: port in
          /nms-system:system/nms-system:syslog-servers/nms-system:server[nms-system:host='10.192.10.10']
          /nms-system:port: \"WrongPort\" is not a valid value.",
          "error-urlpath": "/api/config/system/syslog-servers",
          "error-tag": "malformed-message"
        }
        ]
    }
}
```

## Custom APIs that Interact with YANG and Relational Data Models

The path for APIs that interact with the YANG model and with relational data models starts with /vnms. These APIs return the following standard HTTP status codes. For soma error scenarios, these APIs also return the application error code.

| Status | Description |
|---|---|
| 200 | OK (success) |
| 401 | Unauthorized |
| 404 | Not found |
| 405 | Method not allowed |
| 415 | Unsupported media type |
| 500 | Internal error |

The following examples show calls to YANG and relational data APIs and the values returned:

- Success code

```
curl -i -k -X GET 'https://director-ip-address:9182/vnms/sdwan/workflow/devices?offset=0&limit=1' -H
"Accept: application/json" -H "Content-Type: application/json" -u "Administrator:Versa@123"
HTTP/1.1 200 OK
curl -i -k -X PUT 'https://director-ip-address:9182/vnms/sdwan/workflow/devices
/device/testBranch1-1' -H "Accept: application/json" -H "Content-Type: application/
json" -u "Administrator:Versa@123" -d '{"versanms.sdwan-device-workflow":{"deviceName":"testBranch1-1",
"siteId":"103", "orgName":"Customer1","serialNumber":"sr103", "deviceGroup":"dg2",
"locationInfo":{"country":"IN","longitude":"78.96288","latitude":"20.593684"}}}'
HTTP/1.1 200 OK
```

- Error code

```
curl -i -k -X PUT 'https://director-ip-address:9182/vnms/sdwan/workflow/devices/device/
testBranch1-1' -H "Accept: application/json" -H "Content-Type: application/json"
-u "Administrator:Versa@123" -d '{"versanms.sdwan-device-workflow":{"deviceName":"testBranch1-1",
"siteId":"104","orgName": "Customer1","serialNumber":"sr103", "deviceGroup":"dg2", "locationInfo":
{"country":"IN","longitude":"78.96288","latitude":"20.593684"}}}'
HTTP/1.1 500 Server Error
   {
      "error":{
         "http_status_code":500,"code":0,
         "message":"Global Device Id 104 is already in use.","more_info":
         "http://nms.versa.com/errors/null"
      }
   }

curl -i -k -X GET 'https://director-ip-address:9182/vnms/sdwan/workflow/devices?offset=0&limit=1'
-H "Accept: application/json" -H "Content-Type: application/json" -u "Administrator:Versa@1234"
HTTP/1.1 401 Unauthorized
   {
      "error":{
         "http_status_code":401,"code":4001,"message":"Unauthenticated",
         "description":"Invalid user name or password.","more_info":
         "http://nms.versa.com/errors/4001"
      }
   }
```

## OAuth APIs for Managing OAuth Tokens

The OAuth APIs for managing OAuth tokens use port 9183. The path for token management APIs starts with /auth.

These APIs generate the following status and error codes:

| Status | Description |
|---|---|
| 200 | OK (success) |
| 201 | Created |
| 202 | Accepted |
| 204 | No content |
| 400 | Bad request |

| Status | Description |
|---|---|
| 401 | Unauthorized |
| 405 | Method not allowed |
| 415 | Unsupported media type |
| 500 | Internal error |

The following examples show calls to OAuth token APIs and the values returned:

- Success code

```
curl -i -k -X POST 'https://director-ip-address:9183/auth/token' -H "Accept:
application/json" -H "Content-Type: application/json" -d '{"client_id":
"voae_rest", "client_secret": "asrevnet_123", "username": "Administrator",
"password":"Versa@123","grant_type": "password" }'
HTTP/1.1 200 OK
```

- Error code

```
curl -i -k -X POST 'https://director-ip-address:9183/auth/token' -H "Accept:
application/json" -H "Content-Type: application/json" -d '{"client_id":
"WrongClient", "client_secret": "asrevnet_123", "username": "Administrator",
"password":"Versa@123","grant_type": "password" }'
HTTP/1.1 401 Unauthorized
   {
      "error":"invalid_client","error_description":"Client authentication failed
      (e.g., unknown client, no client authentication included, or unsupported
      authentication method)."
    }

curl -i -k 'https://director-ip-address:9183/vnms/sdwan/
workflow/devices?offset=0&limit=1' -X GET -H "Accept: application/json"
-H "Content-Type: application/json" -H "Authorization:
Bearer 0e4a735f0bf1a4fe889181c05596a048e622c2b4a0d3433543e841dadc7f0a86"
HTTP/1.1 401 Unauthorized
   {
      "error":"invalid_token","error_description":"Invalid access token passed in the request"
   }
```

# How To Use the REST APIs

This section describes how to start using the Director REST APIs.

## Perform Basic Authentication

A Director node supports REST API authentication using HTTP basic authentication with username and password. For example:

```
curl -i -k -X GET 'https://director-ip-address:9182/vnms/ organization/orgs?offset=0&limit=25'
-H "Accept: application/json" -H "Content-Type:application/json" -u Administrator:versa123
```

## Authenticate with OAuth Tokens

A Director node supports REST API authentication using OAuth tokens. For example:

```
curl -i -k 'https://director-ip-address:9183/vnms/organization/
orgs?offset=0&limit=3' -X GET -H "Accept: application/json"
-H "Content-Type: application/json" -H "Authorization:
Bearer 0e4a735f0bf1a4fe889181c05596a048e622c2b4a0d3433543e841dadc7f0a86"
```

## Manage OAuth Tokens

Follow these steps to create a new OAuth client and access token, and to access the Versa Director REST APIs using the access token. In the procedure, replace the IP address in the URL with the IP address of your Director node.

1. Obtain an access token from the voae_rest client. Make a note of the client ID and the client secret.

   ```
   curl -i -k -X POST 'https://director-ip-address:9183/auth/token' -H "Accept: application/json"
   -H "Content-Type: application/json" -d '{"client_id": "voae_rest", "client_secret": "asrevnet_123",
   "username": "Administrator", "password":"versa123","grant_type": "password" }'
   ```

2. Create a new OAuth client using the access token you obtained in Step1. Specify the appropriate values for name, max tokens, and token validity. Note that only Grant_type: client_credentials can be used to manage OAuth clients.

   ```
   curl -i -k -X POST 'https://director-ip-address:9183/auth/admin/
   clients' -H "Accept: application/json" -H "Content-Type: application/json"
   -H "Authorization: Bearer 0c900b8fa6fca010537c1f1121b5b3b53fa02f68b233b00b742aad7c143fe867"
   -d '{
       "name": "My-OAuth-client",
       "description": "My OAuth client",
       "expires_at": "",
       "client_secret_expires_at": "",
       "max_access_tokens": 10,
       "max_access_tokens_per_user": 50,
       "access_token_validity": 900,
       "refresh_token_validity": 86400,
       "allowed_grant_types": [
           "password",
           "refresh_token",
           "client_credentials"
       ],
   ```

```
    "allowed_source_client_address": {
        "source_type": "ANYWHERE",
        "ip_address_list": []
    },
    "enabled": true,
    "software_id": "",
    "software_version": "",
    "contacts": [],
    "redirect_uris": []
}'
```

3. Create one or more new tokens using the new OAuth client. The max_access_tokens_per_user value defines the maximum number of tokens that you can create per user.

```
"Content-Type: application/json"
-d '{
    "client_id": "44002390BCBB0017AFD85ED803871F26",
    "client_secret": "6830fb221368f05b629d0887ece9f6f8",
    "username": "Administrator",
    "password":"versa123",
    "grant_type": "password"
}'
```

4. If necessary, update the OAuth client setting using the token obtained in Step 3. For example:

```
curl -i -k -X PUT 'https://<VERSA DIRECTOR-IP>:9183/auth/admin/
clients/44002390BCBB0017AFD85ED803871F26'
-H "Accept: application/json"
-H "Content-Type: application/json"
-H "Authorization: Bearer 0e4a735f0bf1a4fe889181c05596a048e622c2b4a0d3433543e841dadc7f0a86"
-d '{
    "name": "My-OAuth-client2",
    "description": "My OAuth client 2",
    "expires_at": "",
    "client_secret_expires_at": "",
    "max_access_tokens": 10,
    "max_access_tokens_per_user": 60,
    "access_token_validity": 900,
    "refresh_token_validity": 86400,
    "allowed_grant_types": [
        "password",
        "refresh_token",
        "client_credentials"
    ],
    "allowed_source_client_address": {
        "source_type": "ANYWHERE",
        "ip_address_list": []
    },
    "enabled": true,
    "software_id": "",
    "software_version": "",
    "contacts": [],
    "redirect_uris": []
}'
```

5. Use the OAuth token to access any REST APIs:

```
curl -i -k 'https://director-ip-address:9183/vnms/organization/
orgs?offset=0&limit=3' -X GET -H "Accept: application/json" -H "Content-Type: application/json"
-H "Authorization: Bearer 0e4a735f0bf1a4fe889181c05596a048e622c2b4a0d3433543e841dadc7f0a86"
```

6. Refresh the OAuth client when the access token expires. Note that the scheduler, which runs daily, clears expired tokens.

```
curl -i -k 'https://director-ip-address:9183/auth/refresh' -X POST -H "Accept: application/json"
-H "Content-Type: application/json"
-d '{
    "grant_type": "refresh_token",
    "client_id": "44002390BCBB0017AFD85ED803871F26",
    "client_secret": "6830fb221368f05b629d0887ece9f6f8",
    "refresh_token":"7f3be79b1b0474bb0f71ac4ffc3e1241dcb90a079b5e52913b80ff82e795eb0f"
}'
```

7. Revoke the access token as necessary:

```
curl -i -k 'https://director-ip-address:9183/auth/revoke' -X POST
-H "Accept: application/json"
-H "Authorization: Bearer b958e77b4333fb453aa73f2c19c4d7443399983062da8e221b2aab1b01f33ab8"
```

   The response should be:

```
{"status":"success"}
```

## REST API Ports

Director nodes use the following ports for authentication:

- 9182—For REST APIs with HTTPS basic authentication
- 9183—For REST APIs based on an OAuth token

## REST API HTTP Request Headers

Director REST APIs have the following HTTP request headers:

| API Type | Header |
|---|---|
| Normal APIs | Content-Type: application/json<br>Accept: application/json<br>Authorization: BasicUsername:Password |
| OAuth APIs | Content-Type: application/json |

| API Type | Header |
| --- | --- |
| | Accept: application/json<br>Authorization: Bearer {a valid access token} |

## Use the Browser Developer Tool To Obtain API and Payload from the GUI

It is recommended that you use the Google Chrome browser to obtain APIs from the Director GUI, because it is easier to find Director API calls. To find an API call, navigate to a specific page in the Director GUI, perform a create, update, fetch, or delete action, and look for corresponding API details, for instance, request URL (API), request method, status code, and payload in the Network section of the developer tools.

This section explains how to use a browser developer tool to obtain the Post, Put, Get and Delete APIs for organization workflows and networks.

## Obtain Organization APIs

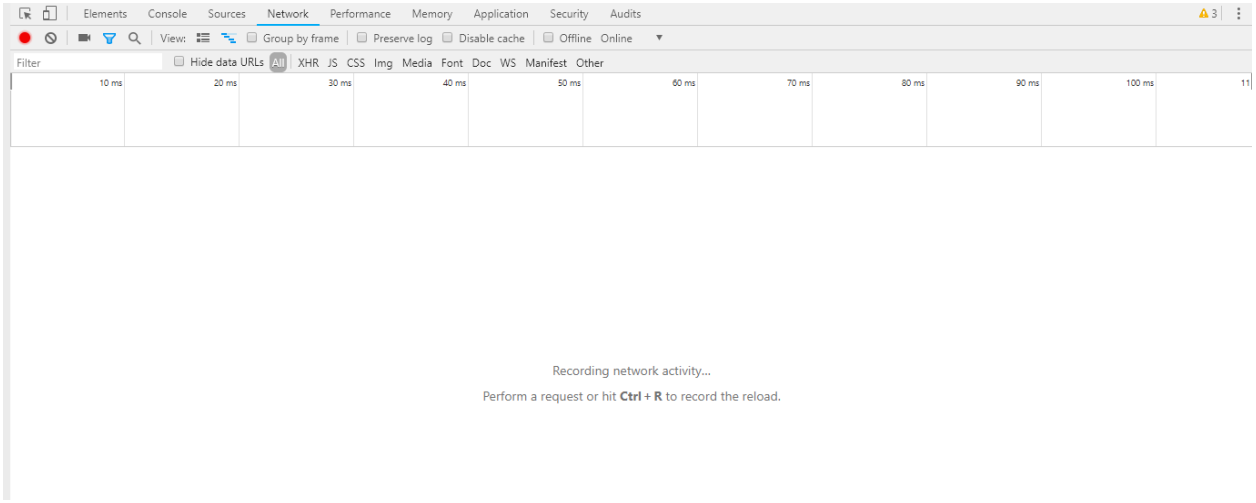
An organization is an entity for which the subscription plan is defined. VOS devices support both single and multiple tenants.

Single tenancy is a single organization with no child organizations, and multitenancy is a parent-child organization hierarchy. In multitenancy, hardware resources are shared, thereby saving on setup time and operational costs. For example, when network resources and other hardware resources are configured for a parent organization, these hardware resources are shared with the child organizations. Similarly, subscription plans that are associated with the parent organization are automatically inherited by the child organizations.

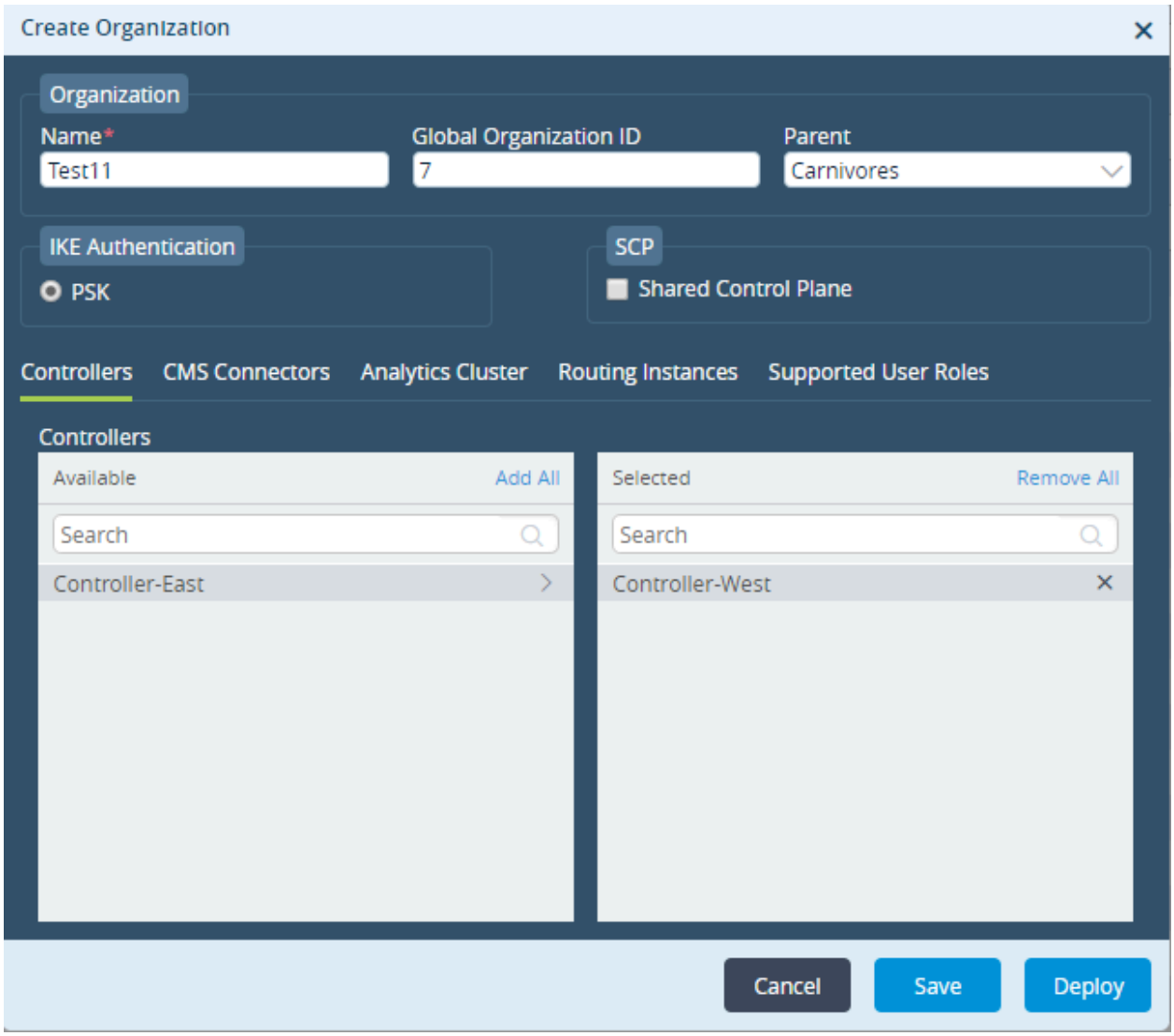
To obtain the API and payload for creating and managing the organization:

1. Right-click the GUI and select Inspect**.**

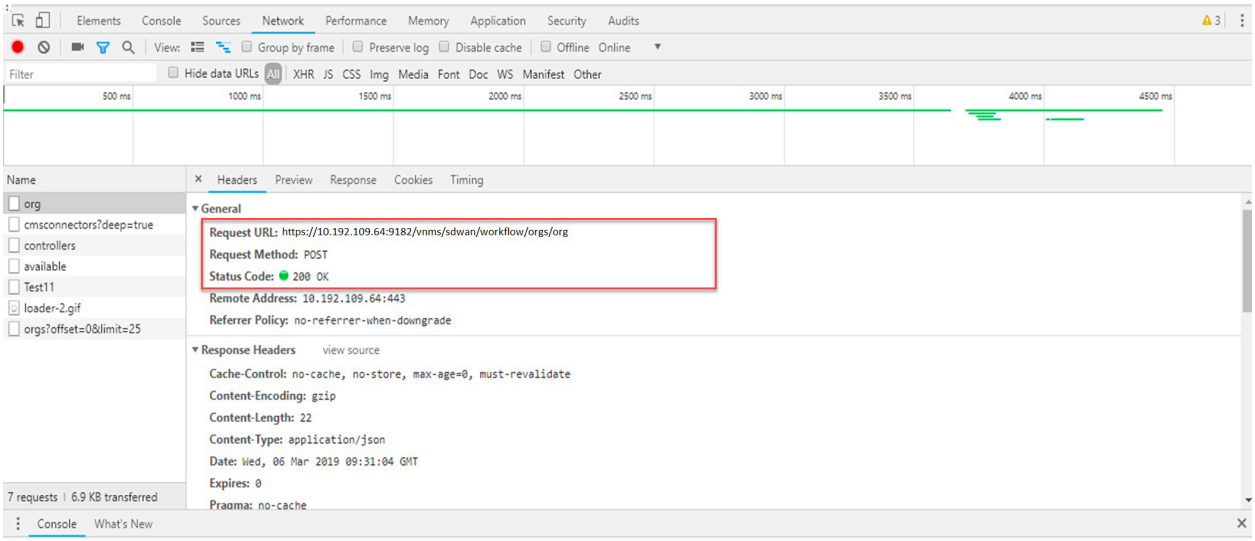

2. Click the Network tab.



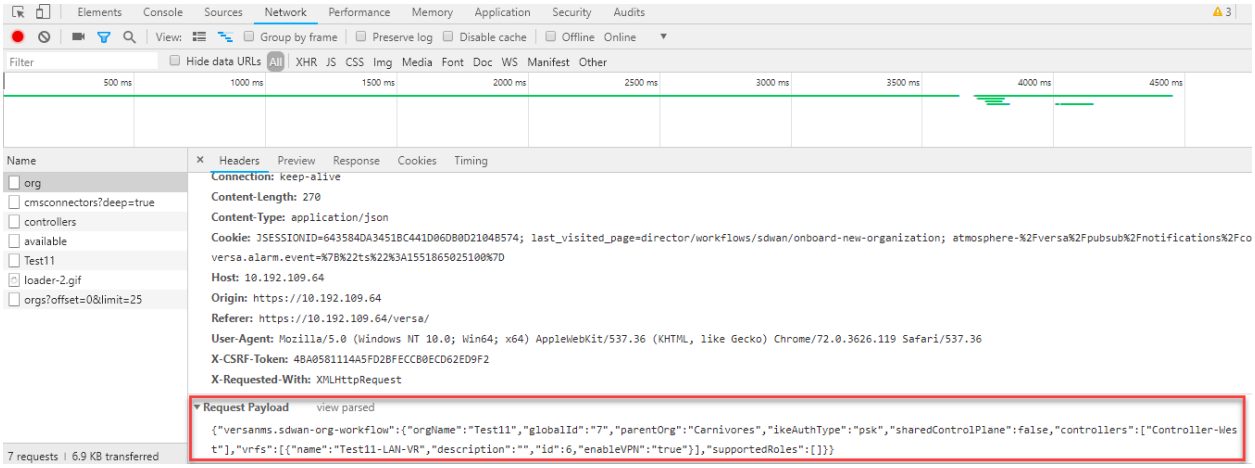

3. On the Director node, select the Workflows tab in the top menu bar, and then select Infrastructure > Organizations in the left menu bar. Click the Add icon to create the organization, and then click Deploy.



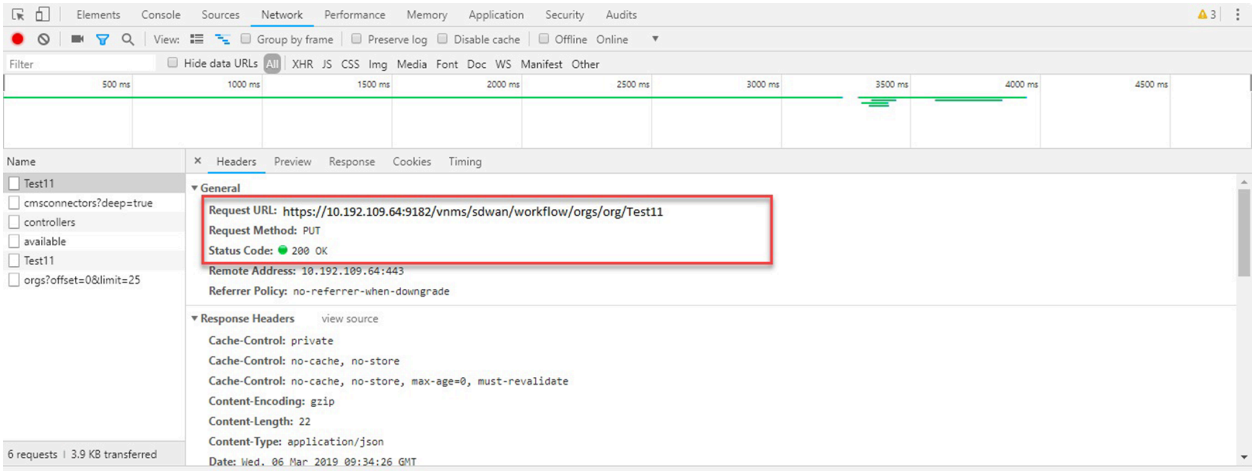

The following Post API is used to create the organization:



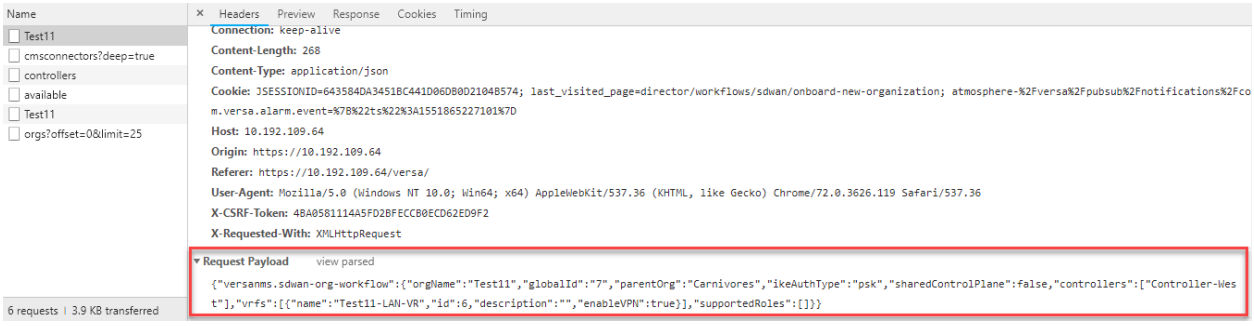

The following shows the API payload:



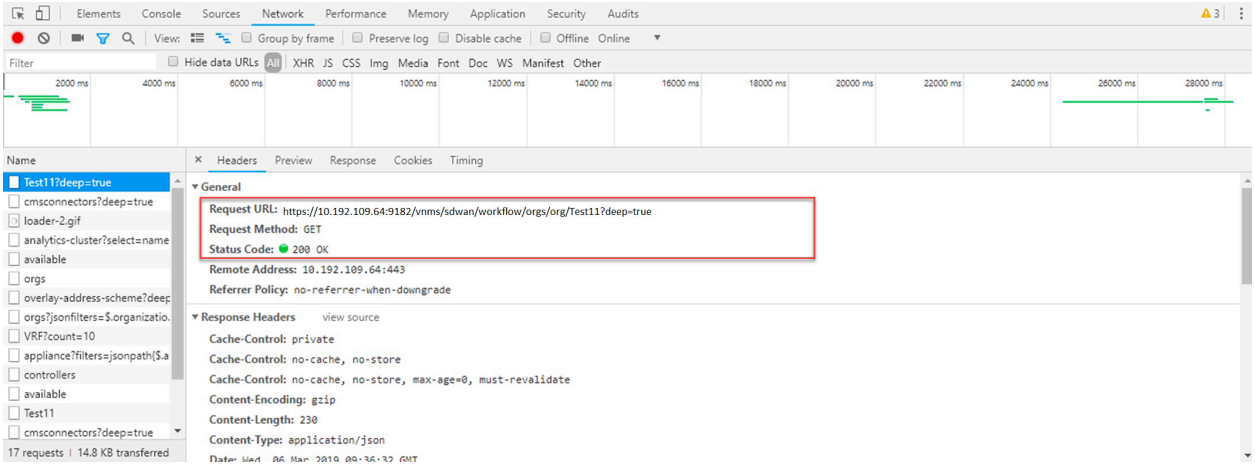

4. To obtain the Put API, open the organization, here, Test11, and click Deploy. The following Put API is used to update the organization.



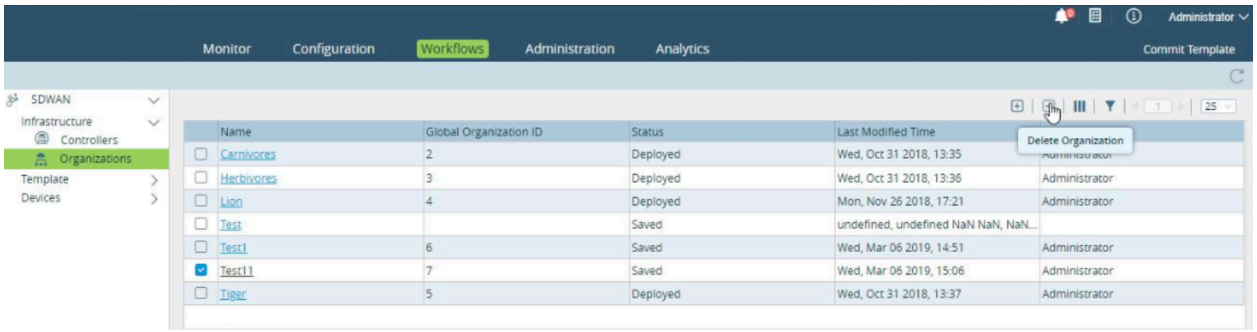

The following shows the API payload:



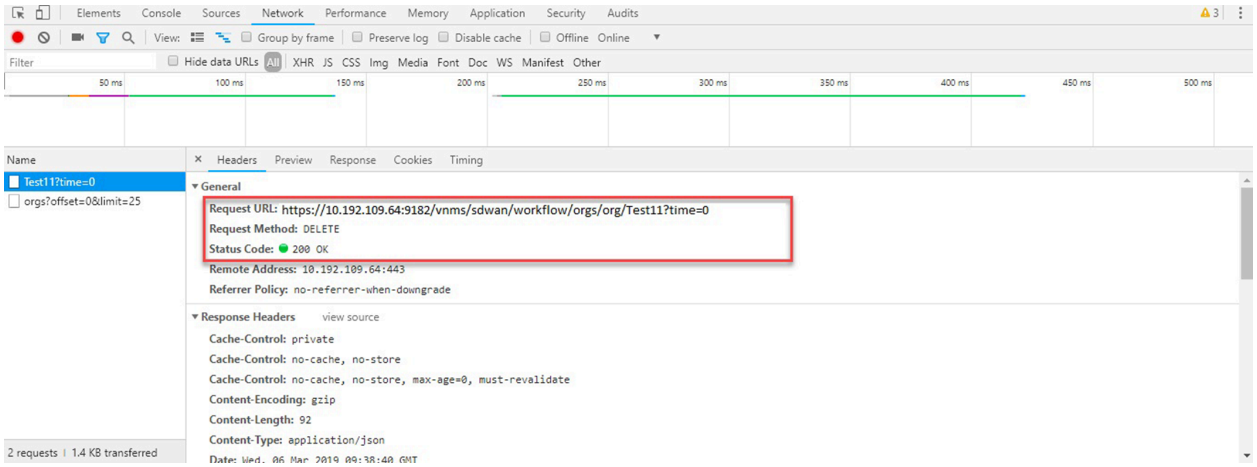

5. To obtain the Get API, open the organization, here, Test11, and click Save. The following Get API is used to fetch the organization.




6. To obtain the Delete API, select the Workflows tab in the top menu bar, select Infrastructure > Organizations in the left menu bar, select the Test11 organization, and click the Delete icon.




The Delete API is used to delete an organization:



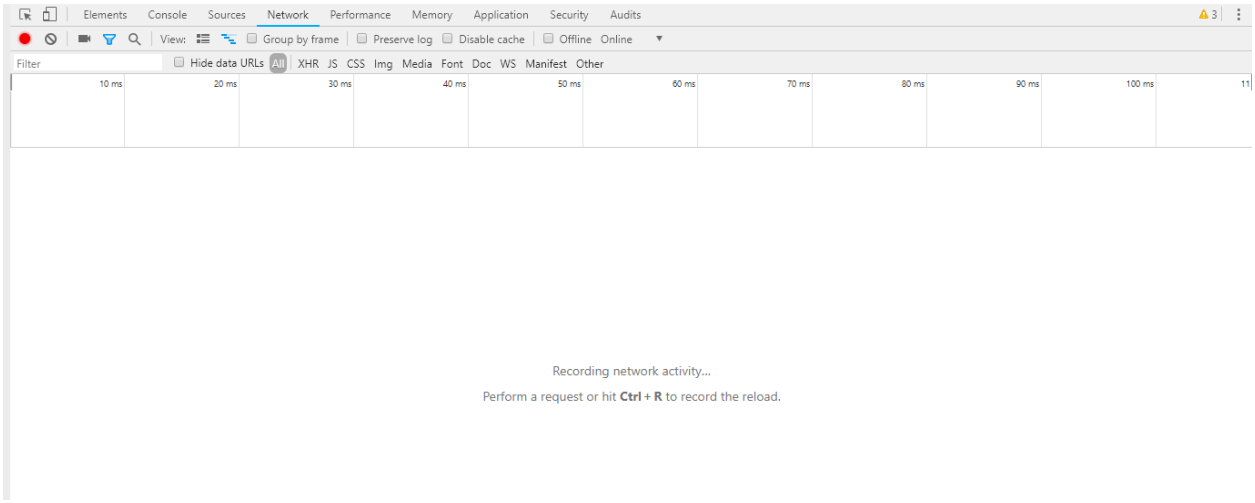

## Obtain Network APIs

A network is a collection of similar interfaces that run on similar routing instances. To help manage multiple interfaces, you can create a network by combining such interfaces. For example, you can create a network with VNI interface that runs on the BGP routing protocol. One interface can be a part of only one network.

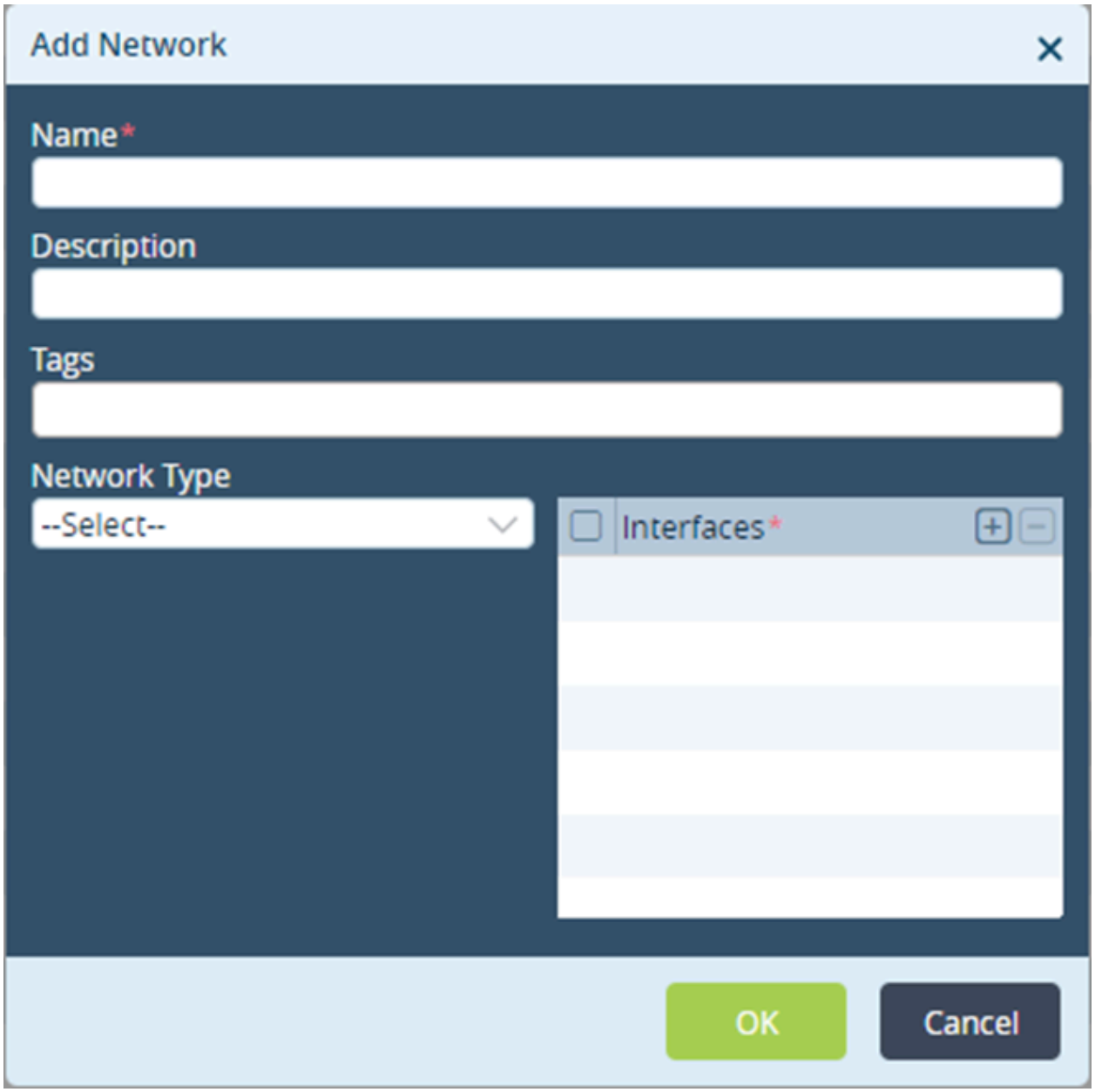
To obtain the API and payload to create and manage a network:

1. Right-click the GUI and select Inspect**.**

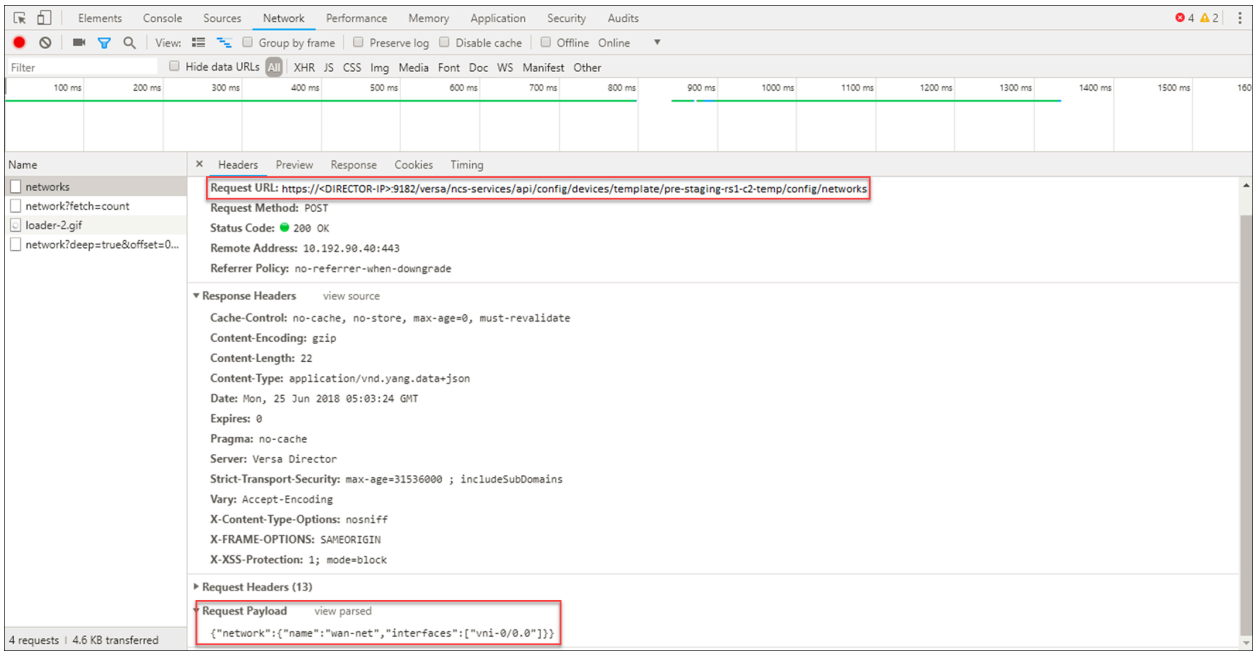

2. Click the Network tab.



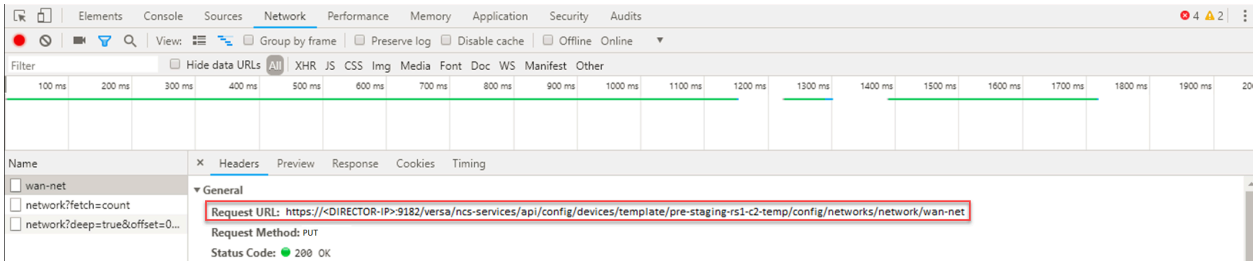

3. On the Director node, select the Configuration tab in the top menu bar, and then select Provider > Device Templates > Networks in the left menu bar. Click the Add icon to create a network, and then click OK.



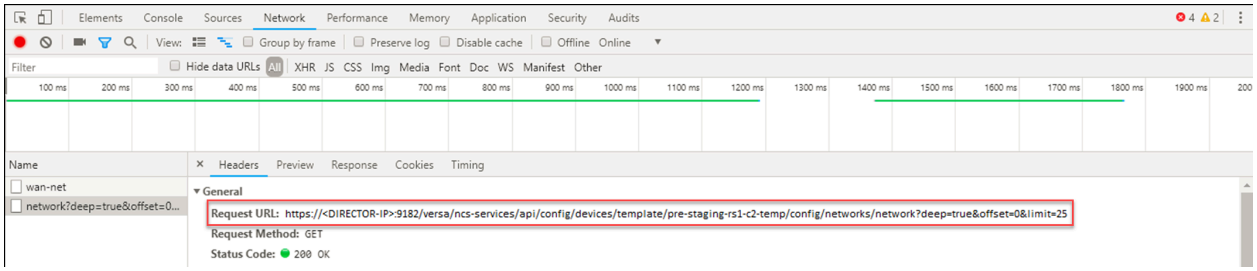

The following Post API is used to create the network.



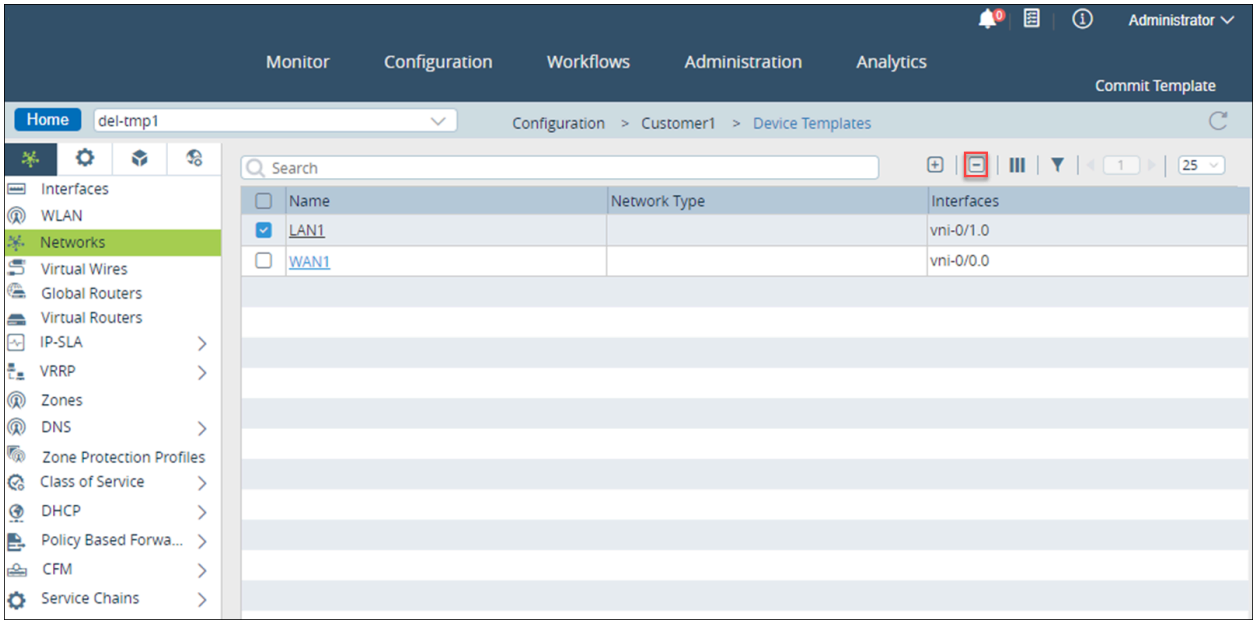

4. To obtain the Put API, which is used to update the network, edit the network.



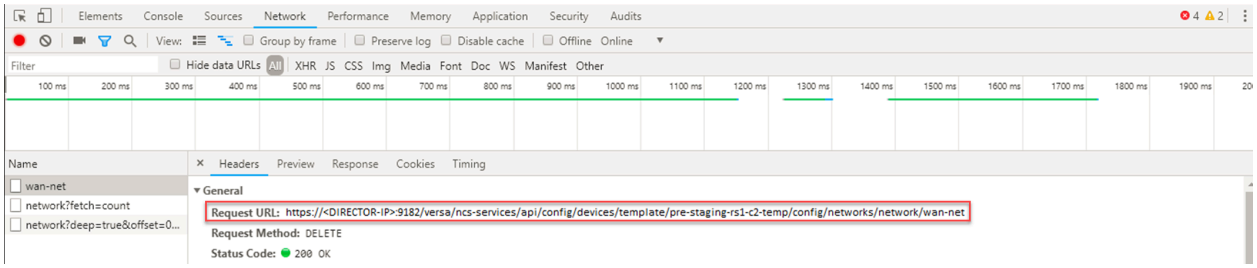

5. To obtain the Get API, which is used to fetch network information, edit the network.




6. To delete a configured network, select the network and click the Delete icon.




The Delete API is used to delete a network.




# Status and Error Responses

The Director APIs support all HTTP status and error codes and OAuth error codes.

## Error Codes

A Director node uses the following HTTP error codes:

| Error Code | Description |
|---|---|
| 4001 | UNAUTHENTICATED |
| 4002 | INVALID_POWER_APPLIANCE_ACTION |
| 4003 | INVALID_SLOT_NO |

| Error Code | Description |
|---|---|
| 4004 | INVALID_APPLIANCE_UUID |
| 4005 | INVALID_CMS_ID |
| 4007 | DUPLICATE_REQUEST |
| 4009 | COMPLETE_ERROR_STACK |
| 4010 | GROUP_NOT_FOUND |
| 4011 | USER_NOT_FOUND |
| 4012 | GROUP_ALREADY_EXIST |
| 4013 | USER_ALREADY_EXIST |
| 4014 | INVALID_TYPE |
| 4015 | NO_DATA_FOUND |
| 4016 | ORG_NOT_FOUND |
| 4017 | INVALID_REQUEST |
| 4018 | IN_USE |
| 4019 | VPEL_SCRIPT_NOT_FOUND |
| 4020 | REFERENCES_IN_USE |
| 4021 | SUB_INTERFACE_FOUND |
| 4022 | INVALID_ACCESS_TOKEN |
| 4023 | INVALID_CLIENT_ID |
| 4024 | INVALID_CLIENT_SECRECT |
| 4025 | ACCESS_TOKEN_EXPIRED |
| 4026 | INVALID_REFRESH_TOKEN |
| 4027 | NO_CLIENT_ID |
| 4028 | EXPIRED_REFRESH_TOKEN |
| 4029 | DATABASE_OPERATION_ERROR |
| 4030 | DATABASE_CONSTRAINTS_UNRESOLVED |

| Error Code | Description |
|---|---|
| 4031 | INVALID_CMS_OPERATION |
| 4032 | CMS_ORG_DELETE_FAILED_ALL |
| 4033 | CMS_ORG_DELETE_FAILED_PARTIAL |
| 5000 | INTERNAL_SERVER_ERROR |

## OAuth Error Codes

The Director REST API returns HTTP OAuth status codes in the response body. The status code includes a JSON XML object that is based on the Accept HTTP request header value and that describes the error.

The XML object contains two fields:

- error—Error code, which is a single string
- error_description—Additional text description about the error, which you can use for debugging

The following is an example of a JSON error response:

```
{ "error":
   "unsupported_grant_type",
   "error_description": "grant type is not supported, only password and refresh_token grant type supported."
}
```

The following table describes the HTTP OAuth errors.

| Error | Description | HTTP Status | Reason or Action To Take |
|---|---|---|---|
| Get Access Token: https://*director-ip-address*:9183/auth/token | | | |
| • invalid_client | Client authentication failed. Possible reasons are unknown client, no client authentication included, and unsupported authentication method. | 401 | Verify that client_id and client_secret are correct. |
| • invalid_grant | Invalid username or password. | 401 | Verify that username and password are correct. |

| Error | Description | HTTP Status | Reason or Action To Take |
|---|---|---|---|
| • invalid_request | Missing parameter: grant_type | 400 | Request did not contain a grant type, or request is missing a required parameter. |
| • unsupported_grant_type | Grant type is not supported. Only password and refresh_token are supported | 400 | Verify that the correct grant_type is passed in the request. |
| Refresh Access Token: https://*director-ip-address*:9183/auth/refresh | | | |
| • invalid_client | Client authentication failed. Possible reasons are unknown client, no client authentication included, and unsupported authentication method | 401 | Verify that client_id and client_secret are correct. |
| • invalid_refresh_token | Invalid refresh token passed in the request. | 401 | Verify that the correct refresh_token value is passed in the request. |
| • invalid_request | Missing parameter: grant_type | 400 | Request did not contain a grant typ, or request is missing a required parameter. |
| • unsupported_grant_type | Grant type is not supported. Pnly password and refresh_token grant type are supported | 400 | Verify that thee correct grant_type is passed in the request. |
| Revoke Access Token: https://*director-ip-address*:9183/auth/revoke | | | |

| Error | Description | HTTP Status | Reason or Action To Take |
|---|---|---|---|
| • invalid_request | Missing access token | 400 | Access token is missing in request headers. |
| • invalid_token | Invalid access token passed in the request | 401 | Verify that the correct access_token value is passed in the request. |
| Use Access Token To Invoke Director APIs | | | |
| • access_forbidden | User is not authorized to perform the action. | 401 | Required permission to execute the action was not been granted. |
| • access_token_expired | Expired access token passed in the request. | 401 | Access token passed in the request is expired. |
| • client_expired | Client has expired. | 401 | Verify the client expiration value. |
| • client_ip_not_allowed | Request from client IP address is not allowed. | 401 | Check client configuration to verify that request is allowed from client's IP address. |
| • client_secret_expired | Client secret has expired. | 401 | Verify the client secret expiration value. |
| • existing_client_found | Existing client with same name found. | 500 | Client with specified name already exists. |
| • invalid_client | Client authentication failed. Possible reasons are unknown client, no client | 401 | Verify that client_id and client_secret are correct. |

| Error | Description | HTTP Status | Reason or Action To Take |
|---|---|---|---|
| | authentication included, and unsupported authentication method | | |
| • invalid_registration_token | Invalid registration token passed in the request. | 401 | Client registration token is invalid. |
| • invalid_request | Missing access token. | 400 | Access token is missing in request headers. |
| • invalid_token | Invalid access token passed in the request. | 401 | Verify that correct access_token value is passed in the request. |
| • registration_token_disabled | Registration token is disabled. | 401 | Client registration token has been disabled. |
| • registration_token_expired | Client registration token has expired. | 401 | Client registration token has expired. |
| • user_credentials_not_matched | User credentials for issuing access_token request have changed. | 401 | Request a new access token. |

## Supported Software Information

Releases 20.2 and later support all content described in this article.

## Additional Information

Access Versa Director REST APIs Using Swagger UI
Troubleshoot REST API Login Issues
Versa Analytics REST API Overview