vRealize Network Insight API Guide

vmware®

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

If you have comments about this documentation, submit your feedback to

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

About This Programming Guide

1

The vRealize® Network Insight™ API Programming Guide provides information about the VMware vRealize Network Insight REST APIs, including how to use the Representational State Transfer (REST) API resources, authenticate, and construct REST API calls.

Intended Audience

This information is intended for administrators and programmers who want to configure and manage vRealize Network Insight programmatically using the vRealize Network Insight REST API. The guide focuses on common use cases.

VMware Technical Publications Glossary

VMware Technical Publications provides a glossary of terms that might be unfamiliar to you. For definitions of terms as they are used in VMware technical documentation, go to

`http://www.vmware.com/support/pubs`

Understanding the REST APIs 2

You can use APIs to automate workflows in vRealize Network Insight. The APIs follow the REST style and is available to all licensed users.

This chapter includes the following topics:

Overview of Rest APIs

The clients of vRealize network Insight API implement a REST workflow making HTTP requests to the server and retrieving the information they need from the server's responses.

About REST

REST, an acronym for Representational State Transfer, describes an architectural style characteristic of applications that use the Hypertext Transfer Protocol (HTTP) to exchange serialized representations of objects between a client and a server. In the vRealize Network Insight API, these representations are JSON documents.In a REST workflow, object representations are passed back and forth between a client and a server with the explicit assumption that neither party need know anything about an object other than what is presented in a single request or response. The URLs at which these documents are available often persist beyond the lifetime of the request or response that includes them.

vRealize Network Insight API Reference is available at:

`https://code.vmware.com/apis/224/vrni`

The API Reference is also available in the product at:

`https://vrni.`*`your_domain`*`.com/doc-api/index.html`

REST API Workflows

Applications that use a REST API send HTTP requests. A script or other higher-level language runs the HTTP requests to make remote procedure calls. These procedure calls create, retrieve, update, or delete objects that the API defines. In the vRealize Network Insight REST API, these objects are defined by a collection of JSON schemas. The operations themselves are HTTP requests, and so are generic to all HTTP clients. To write a REST API client application, you must understand only the HTTP protocol, and the semantics of JSON, the transfer format that the vRealize Network Insight API uses. To use the API effectively in such a client, you must become familiar with the following concepts.

- The set of objects that the API supports, and what they represent.
- How the API represents these objects.
- How a client refers to an object on which it wants to operate. The API reference includes a complete list of API requests and model objects.

API Request

The following HTTP headers are typically included in API requests:

Table 2-1. API Request

Request Header	Description
Authorization	All requests must include an Authorization header with the authentication token generated from vRealize Network Insight auth API.
Content-Type	Requests that include a body must include HTTP Content-Type: application/json header.

Request Body

Ensure that the content in the request body conforms to the type constraint in the model objects in API Reference

API Response

vRealize Network Insight uses conventional HTTP response codes to indicate the success or failure of an API request.

- 2xx range indicates success.
- 4xx range indicates an error when the information provided is incorrect. For example, a required parameter was omitted.
- 5xx range indicates an error with vRealize Network Insight server. However, these errors are rare.

HTTP Response Codes

An API client can expect a subset of HTTP status codes in a response.

Table 2-2. HTTP Response Codes

Status Code	Status Description
200 OK	The request is valid and was completed. The response includes a document body
201 Created	The request is valid. The requested object was created.
204 No Content	The request is valid and was completed. The response does not include a body.
400 Bad Request	The request body is malformed, incomplete, or otherwise invalid.
401 Unauthorized	Login failed or authentication token has expired.
403 Forbidden	The user is not authenticated or does not have adequate privileges to access one or more objects specified in the request.
404 Not Found	The object specified in the request cannot be found
405 Method Not Allowed	The HTTP method specified in the request is not supported for this object.
415 Unsupported Media Type	The resource identified by the request does not support a request of the specified Content-Type and HTTP method.
429 Too Many Requests	A client has sent too many requests or multiple clients are sending too many simultaneous requests and the server is unable to process them due to rate limits. To work around the problem, try sending the request again later.
500 Internal Server Error	The request was received but cannot be completed because of an internal error on the server.
503 Service Unavailable	The server is unable to handle the request due to a temporary condition such as resource exhaustion or server maintenance

REST API Services

This topic provides a summary of REST API services and their functions.

Table 2-3. REST API Services

Service	Description
Authentication	Acquire or delete an authentication token based on the user credentials provided
Data Source Management	Retrieve, update, add, and delete various data sources supported by vRNI
Entities	Retrieve entities from the vRealize Network Insight inventory
Applications	Create Applications and Tiers based on virtual machine membership criteria and/or IP address/subnet criteria.
Micro Segmentation	Determine the firewall rules recommended by vRealize Network Insight for the different tiers, security groups, and applications.
Search	Search vRNI entities.

Rest API Authentication and Authorization

vRealize Network Insight supports token based authentication. Tokens are non-modifiable identifiers returned by the system when the user has successfully authenticated using valid credentials.

vRealize Network Insight requires API requests to be authenticated. The first step in this workflow is to obtain an authentication token. To obtain an authentication token, the login request supplies the user credentials. In this example, the user is logging in to a vRealize Network Insight instance with URL `https:/vrni.example.com/`.

Acquire an authentication token

Prerequisites

- Secure a channel between the web browser and the vRealize Network Insight server. Open a browser and enter the URL of a vRealize Network Insight instance such as `https://vrni.example.com`.

  The system warns that your connection is not private. Click through to confirm the security exception and establish an SSL handshake
- Verify that you have the login credentials for a user of your vRealize Network Insight instance.

Steps

- POST a request to the authentication URL to acquire a token.

  POST `https://vrni.example.com/api/ni/auth/token`
- Examine the response. A successful request returns an authorization token that you must include in subsequent API requests.

```
Request header:
          POST https://vrni.example.com/api/ni/auth/token
    Content-Type: application/json
    Accept: application/json
    Request body in JSON format:
    {
          "username": "testuser@local",
         "password": "testpassword",
          "domain": {
            "domain_type": "LOCAL"
         }
    }
Response:
    {
            "token": "6QfAhSQ7/ivmvDkHPOEvTw==",
            "expiry": 1508767809240
    }
```

The authorization token is valid for 5 hours after generation. If the token expires, you must regenerate it.

Delete an authorization token

You can also delete a token before it expires using the following http request.

```
Request Header
    DELETE https://vrni.example.com/api/ni/auth/token
    Authorization: NetworkInsight auth-token
```

Use an authentication token

Each API request must contain the authorization token in Authorization Header in the following format:

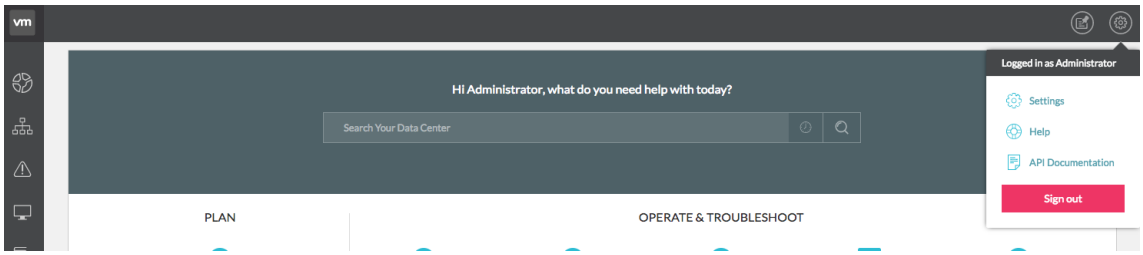
`Authorization: NetworkInsight auth-token`

Using API Explorer

vRealize Network Insight API Explorer is available in product which is used to explore APIs and their responses.

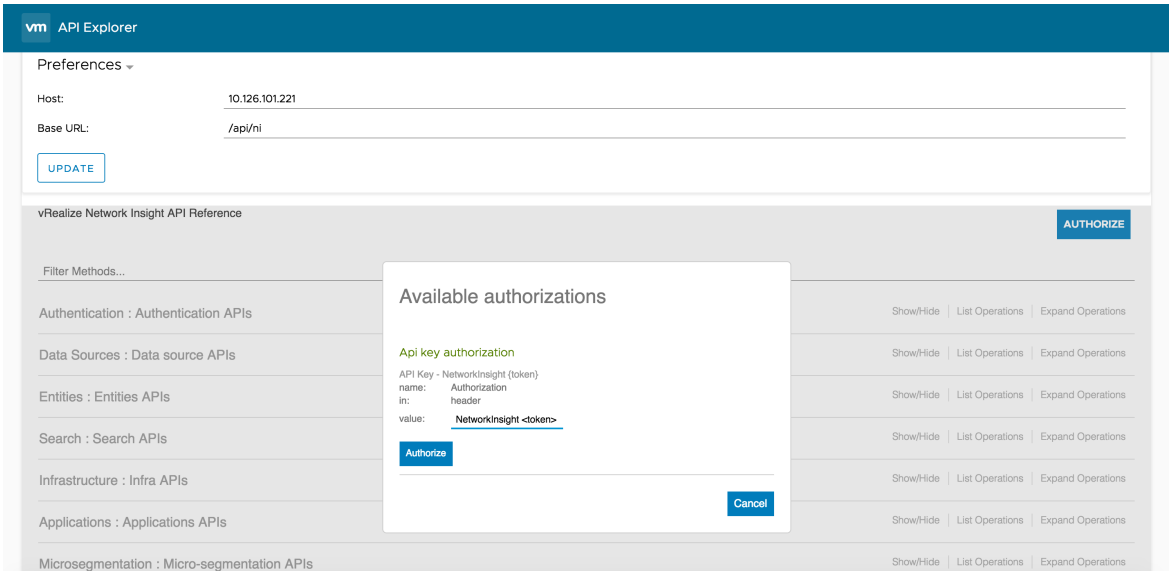
Steps to use API Explorer.

- vRealize Network Insight API Explorer can be accessed from API Documentation link as shown below.



- You should change the Host in Preferences section from "vrni.example.com" to ip address/fqdn of vRealize Network Insight deployment.

- Generate auth token using authentication API.
- Click on the "Authorize" button to enter the auth token as shown below.

- 

- Now you can try any API using "try it out!" button.

Managing Data Sources

3

You can manage data sources using APIs. Data sources can be added, updated, deleted, enabled, or disabled using APIs.

Data Sources URL Prefix

`/data-sources/`

Data sources are grouped on the basis of the data source type. URL Prefix corresponding to data sources types are listed in this table.

Table 3-1. Data Source Type to URL Prefix

Data Source Type	URL Prefix
VCenterDataSource	`/data-sources/vcenters`
NSXVManagerDataSource	`/data-sources/nsxv-managers`
CiscoSwitchDataSource	`/data-sources/cisco-switches`
AristaSwitchDataSource	`/data-sources/arista-switches`
DellSwitchDataSource	`/data-sources/dell-switches`
BrocadeSwitchDataSource	`/data-sources/brocade-switches`
JuniperSwitchDataSource	`/data-sources/juniper-switches`
UCSManagerDataSource	`/data-sources/ucs-managers`
HPOneViewDataSource	`/data-sources/hpov-managers`
HPVCManagerDataSource	`/data-sources/hpvc-managers`
CheckpointFirewallDataSource	`/data-sources/checkpoint-firewalls`
PanFirewallDataSource	`/data-sources/panorama-firewalls`

List Data Sources

Prerequisites

- Get the data source URL for the data source type. For more information, see Table 3-1: DataSourceType to URL Prefix.

Procedure

- Make a GET request to the URL corresponding to the data source type.

```
    Request
        GET https://vrni.example.com/api/ni/data-sources/vcenters
    Response Body
{
    "results": [
        {
            "entity_id": "10000:902:627340223",
            "entity_type": "VCenterDataSource"
        },
        {
            "entity_id": "10000:902:993642840",
            "entity_type": "VCenterDataSource"
        },
        {
            "entity_id": "10000:902:738162743",
            "entity_type": "VCenterDataSource"
        },
        {
            "entity_id": "10000:902:627340998",
            "entity_type": "VCenterDataSource"
        },
        {
            "entity_id": "10000:902:390269772",
            "entity_type": "VCenterDataSource"
        }
    ],
    "total_count": 5
}
```

Fetch Data Source Details

Prerequisites

- Get the data source URL for the data source type. For more information, see Table 3-1: DataSourceType to URL Prefix.
- Get entity ID using the `list` command.

Procedure

- Make a GET request for the specific data source entity ID.

  For example:

```
Request
    GET https://vrni.example.com/api/ni/data-sources/vcenters/10000:902:627340223
Repsonse Body
    {
        "entity_id": "10000:902:993642895",
        "entity_type": "VCenterDataSource",
        "ip": "10.197.17.68",
        "proxy_id": "10000:901:1586035958",
         "nickname": "aa",
        "enabled": true,
         "notes": "ecmp lab aa",
         "credentials": {
            "username": "administrator@vsphere.local",
             "password": ""
        }
    }
```

Update Data Source Credentials

Prerequisites

- Get the data source URL for the data source type. For more information, see Table 3-1: DataSourceType to URL Prefix.
- Get entity ID using the `list` command.

Procedure

- Make a GET request for the specific data source entity ID.
- Make a PUT request with request body after changing the credentials in the response retrieved in step 1.

  For example:

```
Request
    PUT https://vrni.example.com/api/ni/data-sources/vcenters/10000:902:627340223
Request Body
    {
        "entity_id": "10000:902:993642895",
        "entity_type": "VCenterDataSource",
        "ip": "10.197.17.68",
        "proxy_id": "10000:901:1586035958",
         "nickname": "aa",
        "enabled": true,
         "notes": "ecmp lab aa",
         "credentials": {
            "username": "newuser",
```

```
                "password": "newpassword"
            }
        }
    Response Body
        {
            "entity_id": "10000:902:993642895",
            "entity_type": "VCenterDataSource",
            "ip": "10.197.17.68",
            "proxy_id": "10000:901:1586035958",
             "nickname": "aa",
            "enabled": true,
             "notes": "ecmp lab aa",
             "credentials": {
                "username": "newuser",
                 "password": ""
            }
        }
```

Enable a Data Source

Prerequisites

- Get the data source URL for the data source type. For more information, see Table 3-1: DataSourceType to URL Prefix.
- Get entity_id from the `list` command.

Procedure

- To enable a data source, make a POST request.

  For example:

```
    Request
        POST
        https://vrni.example.com/api/ni/data-sources/vcenters/10000:902:627340223/enable
    Repsonse
        200 OK
```

Disable a Data Source

Prerequisites

- Get the data source URL for the data source type. For more information, see Table 3-1: DataSourceType to URL Prefix.
- Get entity_id from the `list` command.

Procedure

- To disable a data source, make a POST request.

  For example:

```
Request
    POST
    https://vrni.example.com/api/ni/data-sources/vcenters/10000:902:627340223/disable
Repsonse
    200 OK
```

Add a Data Source

Prerequisites

- Get the data source URL for the data source type. For more information, see Table 3-1: DataSourceType to URL Prefix.
- You must have IP Address or the FQDN of the data source and its credentials.

Procedure

- Get the proxy id for adding data sources using the infra/nodes APIs.
- Make a POST request conforming to the API specification. For example, to add a vCenter data source:

```
Request
      GET

Request
        POST https://vrni.example.com/api/ni/data-sources/vcenters/
    Request Body
    {
        "ip": "10.197.17.68",
        "fqdn": "",
        "proxy_id": "10000:901:1586035958",
        "nickname": "aa",
        "enabled": true,
        "notes": "ecmp lab aa",
        "credentials":
        {
            "username": "administrator@vsphere.local",
            "password": "password"}
    }

    Response Body
    {
        "entity_id": "10000:902:993642895",
        "entity_type": "VCenterDataSource",
        "ip": "10.197.17.68",
         "proxy_id": "10000:901:1586035958",
         "nickname": "aa",
```

```
        "enabled": true,
        "notes": "ecmp lab aa",
        "credentials": {
           "username": "administrator@vsphere.local",
           "password": ""
        }
    }
```

Delete a Data Source

Prerequisites

- Get the data source URL for the data source type. For more information, see Table 3-1: DataSourceType to URL Prefix.
- Get entity_id from the `list` command.

Procedure

- Make a DELETE request for the specific entity_id.

```
Request DELETE https://vrni.example.com/api/ni/data-sources/vcenters/10000:902:627340223/
Response
   204 No Content
```

Configure SNMP on a Switch Data Source

Prerequisites

- Get the switch data source URL for the data source type. For more information, see Table 3-1: DataSourceType to URL Prefix.
- Get entity_id from the `list` command.

Procedure

- Make a PUT request conforming to the API specification.

```
Request PUT
https://vrni.example.com/api/ni/data-sources/cisco-switches/10000:903:627340223
Request Body
{
        "snmp_enabled": true,
        "snmp_version": "v3",
        "config_snmp_3": {
            "username": "nilesh",
            "authentication_type": "SHA",
            "authentication_password": "authentication_password",
            "privacy_type": "NO_PRIV"
    }

Response Body
{
```

```
            "snmp_enabled": true,
            "snmp_version": "v3",
            "config_snmp_3": {
                "username": "nilesh",
                "authentication_type": "SHA",
                "privacy_type": "NO_PRIV"
        }
```

Tagging IP Addresses

4

IP Addresses/Subnets can be tagged with EAST_WEST/INTERNET tags with this API.

List All Available Tags

- Following API returns all the available tags in system.

```
Request
    GET https://vrni.example.com/api/ni/ip-tags
Response
{
  "tag_ids": [
    "INTERNET",
    "EAST_WEST"
  ]
}
```

Add IP Address/Subnet to a Tag.

- Use the following request to add a subnet or IP address (es) to a tag. In the following example, Subnet and IP addresses are being tagged with EAST_WEST tag.

```
Request
    POST https://vrni.example.com/api/ni/settings/ip-tags/EAST_WEST/add
Request Body
{
  "tag_id": "EAST_WEST",
  "subnets": [
    "192.168.10.0/24"
  ],
  "ip_address_ranges": [
    {
      "start_ip": "192.168.20.1",
      "end_ip": "192.168.20.5"
    }
  ]
}
Response 200 OK.
```

Remove IP Address/Subnet from a Tag

- Use the following request to remove the subnet or IP address (es) for a tag. In the following example, IP address ranges (192.168.20.1-192.168.20.5) are being removed from EAST_WEST tag.

```
Request
    POST https://vrni.example.com/api/ni/settings/ip-tags/EAST_WEST/remove
Request Body
{
  "tag_id": "EAST_WEST",
  "ip_address_ranges": [
    {
      "start_ip": "192.168.20.1",
      "end_ip": "192.168.20.5"
    }
  ]
}
Response 200 OK.
```

Get Tag Details

- Use the following request to get tag details.

```
Request
    GET https://vrni.example.com/api/ni/settings/ip-tags/EAST_WEST
Response
{
  "tag_id": "EAST_WEST",
  "ip_address_ranges": [
    {
      "start_ip": "192.168.20.1",
      "end_ip": "192.168.20.5"
    }
  ]
}
```

Performing Search

5

You can search for entities using the entity type and search criteria. You can define a search criteria using filter expressions based on entity properties as defined in the API reference. Search results return a paginated list of entity IDs that match the filter criteria.

Search Request

Format of the Search Request body:

```
{
  "entity_type": "string",
  "filter": "string",
  "sort_by": {
    "field": "string",
    "order": "ASC"
  },
  "size": 0,
  "cursor": "string",
  "time_range": {
    "start_time": 0,
    "end_time": 0
  }
}
```

Entity Type

In the Entity Type field, use the entity types defined in AllEntityType enum in the API Reference.

Filter Expression

A filter expression defines search criteria in one of the following formats:

- *{field_name} {binary operator} {field_value}*
- *{field_name} {unary_operator}*

Optionally, complex expressions can be designed using {logical operators} and parenthesis (...).

{field_name}	The name of a field in an Object. For example, VirtualMachine has a field named `vendor_id`.
{binary operator}	Used to compare a field to a value creating an expression.
{unary_operator}	Unary operator is used to test the field_name against the operator check. For example, IS SET is used to determine if a field is non NULL.
{field_value}	Represents the value of a field and can either be a number, string , list of numbers or list of strings.
{logical operator}	Logical operators are used to create complex filters. For example: (name = '*my_vm*') OR (vendor_id = '*vm-101*')
parentheses (...)	Parentheses are used to group expressions. They also determine the order in which the components of the expression are evaluated. For example: ((name = '*my_vm*') AND (default_gateway = '*10.1.1.1*'))

Considerations

- You can use spaces in filter expressions.
- Use quotes with strings. You need not use quotes with numbers.

Building Filter Expression

Components of a filter expression are: *FIELD_NAME OPERATOR VALUE*

FIELD_NAME

Field name can be a property defined in the model object as per the API Reference or it can be a property of a related object in the response. For example, a subset of the VirtualMachine object properties appears as follows:


```
{
  "entity_id": "18230:1:1158969162",
  "name": "NSX_Controller_9e80ec74-57ce-4671-8fd7-b5884a997535",
  "entity_type": "VirtualMachine",
  "ip_addresses": [
    {
      "ip_address": "10.197.17.74",
      "netmask": "255.255.255.0",
      "network_address": "10.197.17.0/24"
    }
  ],
  "default_gateway": "10.197.17.1",
  "cluster": {
    "entity_id": "18230:66:1293137396",
```

```
    "entity_type": "Cluster"
  },
  "host": {
    "entity_id": "18230:4:652218965",
    "entity_type": "Host"
  }
}
```

From the preceding model, you can use the following field names:

- Direct primitive properties
  - name
  - default_gateway
- Properties inside the structure.

  For example:
  - ip_addresses.ip_address.
  - host.entity_id
- Properties of a related object.

  For example:
  - host.name
  - cluster.name

Table 5-1. Operators

Operator	Description
=	Equals
!=	Not Equals
>	Greater Than
<	Less Than
>=	Greater Than OR Equals
<=	Less Than OR Equals
LIKE	Value contains the the field value
IN	Value matches one of the field values in list
NOT IN	Value does not match any field value in list
SET	Value is not null
NOT SET	Value is null

FIELD_VALUE

The search value can be a string, integer, list of strings or list of integers.

For example:

- String: 'securitygroup-10'
- Integer : 211

Examples of Search

Sample Search Requests

Find VMs with a specific ip_address, such as 192.168.10.1:

```
Request
    POST https://vrni.example.com/api/ni/search
Request Body
    {
          "entity_type": "VirtualMachine",
          "filter": "ip_addresses.ip_address = '192.168.10.1'",
    }
```

Find all VMs on a host with name 'host-a'

```
Request
    POST https://vrni.example.com/api/ni/search
Request Body
    {
          "entity_type": "VirtualMachine",
          "filter": "host.name = 'host-a'",
    }
```

Find all VMs in the NSX security group with moref 'securitygroup-10'

```
Request
    POST https://vrni.example.com/api/ni/search
Request Body
    {
          "entity_type": "VirtualMachine",
          "filter": "security_groups.vendor_id = 'securitygroup-10'",
    }
```

Find all VMs where the name is 'vm-a' or 'vm-b' or 'vm-c'

```
Request
    POST https://vrni.example.com/api/ni/search
Request Body
    {
          "entity_type": "VirtualMachine",
          "filter": "name IN ('vm-a', 'vm-b', 'vm-c')",
    }
```

Find all VMs where the default gateway is set

```
    Request
        POST https://vrni.example.com/api/ni/search
    Request Body
        {
              "entity_type": "VirtualMachine",
              "filter": "default_gateway is set",
        }
```

Find all VMs in host 'host-a' and security group 'sg-1'

```
    Request
        POST https://vrni.example.com/api/ni/search
    Request Body
        {
              "entity_type": "VirtualMachine",
              "filter": "((security_groups.name = 'sg-1') and (host.name = 'host-a'))",
        }
```

Find all VMs in security group sg-1 and that do not have network address '192.168.10.0/24'

```
    Request
        POST https://vrni.example.com/api/ni/search
    Request Body
        {
              "entity_type": "VirtualMachine",
              "filter": "((security_groups.name = 'sg-1') and (ip_addresses.network_address !=
'192.168.10.0/24'))",
        }
```

Working with Entities

6

Each entity in vRealize Network Insight has an associated entity ID that is unique and an entity type. You can fetch an entity using the entity ID from the URL corresponding its entity type. You can get the list of all entity IDs using a paginated list request.

You can determine the entity ID using a generic search based on the properties of the entity. For more information, see the Performing Search topic.

List All Entities for an Entity Type

Listing all entities of an entity type requires issuing a GET request on the URL corresponding to the entity type. The response contains a paginated list of entities with time corresponding to the version of the entity that has been retrieved. Response contains the cursor that can be used as a query parameter to fetch next page of results. You can add the *size* query parameter to request to set the number of entities that server should send in response. It can be set to a maximum of 10000.

Prerequisite

- Obtain the URL corresponding to the entity type in the search result.

Procedure

- Issue a GET request using the URL. For example, use this GET request to list all VMs in vRealize Network Insight.

```
Request
GET https://vrni.example.com/api/ni/entities/vms/
Response
{
    "results": [
        {
            "entity_id": "10000:1:875335999",
            "entity_type": "VirtualMachine",
            "time": 1508745899
        },
        {
            "entity_id": "10000:1:1206990361",
            "entity_type": "VirtualMachine",
            "time": 1508745899
        },
        ...
```

```
    ],
    "cursor": "MTA=",
    "total_count": 78,
    "start_time": 1508972172,
    "end_time": 1508972172
}
```

- Response contains a paginated list with the cursor to be passed in the next request as a parameter to retrieve the next set of VMs.

Retrieve an Entity

Prerequisite

- Obtain the entity ID of the entity by using a search or list command
- Obtain the URL corresponding to the entity type in the search result. For more information, see Table 3: EntityType to URI.

Procedure

Run a GET request using the entity ID and the URL. This example uses entity_id 1000:1:123121 and entity_type is VirtualMachine.

```
Request
GET https://vrni.example.com/api/ni/entities/vms/1000:1:123121
Response
{
    "entity_id": "10000:1:1207046967",
    "name": "pashan-69-1",
    "entity_type": "VirtualMachine",
    "ip_addresses": [
        {
            "ip_address": "192.168.24.40",
            "netmask": "255.255.255.0",
            "network_address": "192.168.24.0/24"
        }
    ],
    "default_gateway": "192.168.24.10",
    "vnics": [
        {
            "entity_id": "10000:18:928014219",
            "entity_type": "Vnic"
        }
    ],
    "security_groups": [
        {
            "entity_id": "10000:82:1509031646",
            "entity_type": "NSXSecurityGroup"
        }
    ],
    "source_firewall_rules": [
        {
            "rules": [
```

```
                {
                    "entity_id": "10000:87:1310503280",
                    "entity_type": "NSXFirewallRule"
                },
                {
                    "entity_id": "10000:87:1310503311",
                    "entity_type": "NSXFirewallRule"
                }
            ],
            "firewall": {
                "entity_id": "10000:39:306795770",
                "entity_type": "NSXDistributedFirewall"
            },
            "rule_set_type": "NSX_STANDARD"
        }
    ],
    "destination_firewall_rules": [
        {
            "rules": [
                {
                    "entity_id": "10000:87:1310621266",
                    "entity_type": "NSXFirewallRule"
                },
                {
                    "entity_id": "10000:87:1310503280",
                    "entity_type": "NSXFirewallRule"
                }
            ],
            "firewall": {
                "entity_id": "10000:39:306795770",
                "entity_type": "NSXDistributedFirewall"
            },
            "rule_set_type": "NSX_STANDARD"
        }
    ],
    "ip_sets": [
        {
            "entity_id": "10000:84:1256619295",
            "entity_type": "NSXIPSet"
        }
    ],
    "cluster": {
        "entity_id": "10000:66:850307201",
        "entity_type": "Cluster"
    },
    "resource_pool": {
        "entity_id": "10000:79:1730744919",
        "entity_type": "ResourcePool"
    },
    "security_tags": [
        {
            "entity_id": "10000:99:1129991034",
            "entity_type": "SecurityTag"
        }
    ],
```

```
    "layer2_networks": [
        {
            "entity_id": "10000:11:604852557",
            "entity_type": "VxlanLayer2Network"
        }
    ],
    "host": {
        "entity_id": "10000:4:2142603919",
        "entity_type": "Host"
    },
    "vlans": [],
    "vendor_id": "vm-174",
    "vcenter_manager": {
        "entity_id": "10000:8:2048038675",
        "entity_type": "VCenterManager"
    },
    "folders": [
        {
            "entity_id": "10000:81:1937753507",
            "entity_type": "Folder"
        }
    ],
    "datastores": [
        {
            "entity_id": "10000:80:2099650948",
            "entity_type": "Datastore"
        }
    ],
    "datacenter": {
        "entity_id": "10000:105:246393763",
        "entity_type": "VCDatacenter"
    },
    "nsx_manager": {
        "entity_id": "10000:7:935317189",
        "entity_type": "NSXVManager"
    },
    "applied_to_source_rules": [
        {
            "rules": [
                {
                    "entity_id": "10000:87:1310503280",
                    "entity_type": "NSXFirewallRule"
                },
                {
                    "entity_id": "10000:87:1310622351",
                    "entity_type": "NSXFirewallRule"
                }
            ],
            "firewall": {
                "entity_id": "10000:39:306795770",
                "entity_type": "NSXDistributedFirewall"
            },
            "rule_set_type": "NSX_STANDARD"
        }
    ],
```

```
    "applied_to_destination_rules": [
        {
            "rules": [
                {
                    "entity_id": "10000:87:1310621266",
                    "entity_type": "NSXFirewallRule"
                },
                {
                    "entity_id": "10000:87:1310503280",
                    "entity_type": "NSXFirewallRule"
                }
            ],
            "firewall": {
                "entity_id": "10000:39:306795770",
                "entity_type": "NSXDistributedFirewall"
            },
            "rule_set_type": "NSX_STANDARD"
        }
    ],
    "source_inversion_rules": [],
    "destination_inversion_rules": []
}
```


Retrieve an Entity at Specified Time

vRealize Network Insight stores different versions of an entity and creates an image of the entity with a new timestamp when the data changes. A previous version of an entity can be retrieved by passing the time as query parameter in the retrieve entity request. Ensure to provide the time in epoch seconds.

```
https://vrni.example.com/api/ni/entities/vms/10000:1:875335999?time=1508745500
```

Bulk fetch of entities

Bulk fetch of entities can be fetched using a bulk fetch API. The maximum batch size of bulk fetch is 1000 entities.

```
POST https://vrni.example.com/api/ni/entities/fetch
Request Body
{
 "entity_ids": [
    {
      "entity_id": "10000:4:132029350",
      "entity_type": "Host",
      "time": 1523605300
    },
    {
      "entity_id": "10000:4:132027614",
      "entity_type": "Host",
      "time": 1523605419
    }
```

```
  ]
}

Response Body
{
  "results": [
    {
      "entity_id": "10000:4:132029350",
      "entity_type": "Host",
      "entity": {
        "entity_id": "10000:4:132029350",
        "name": "10.197.52.176",
        "entity_type": "Host",
        "vmknics": [
          {
            "entity_id": "10000:17:771614149",
            "entity_type": "Vmknic"
          }
        ],
        "vcenter_manager": {
          "entity_id": "10000:8:824493708",
          "entity_type": "VCenterManager"
        },
        "vm_count": 4,
        "datastores": [
          {
            "entity_id": "10000:80:1910974952",
            "entity_type": "Datastore"
          },
          {
            "entity_id": "10000:80:1910975665",
            "entity_type": "Datastore"
          },
          {
            "entity_id": "10000:80:1910975696",
            "entity_type": "Datastore"
          },
          {
            "entity_id": "10000:80:889229135",
            "entity_type": "Datastore"
          },
          {
            "entity_id": "10000:80:1910974983",
            "entity_type": "Datastore"
          }
        ],
        "service_tag": "",
        "vendor_id": "host-127",
        "maintenance_mode": "NOTINMAINTENANCEMODE",
        "connection_state": "CONNECTED"
      },
      "time": 1523605300
    },
    {
      "entity_id": "10000:4:132027614",
```

```
      "entity_type": "Host",
      "entity": {
        "entity_id": "10000:4:132027614",
        "name": "10.197.52.178",
        "entity_type": "Host",
        "vmknics": [
          {
            "entity_id": "10000:17:1090151026",
            "entity_type": "Vmknic"
          }
        ],
        "vcenter_manager": {
          "entity_id": "10000:8:824493708",
          "entity_type": "VCenterManager"
        },
        "vm_count": 3,
        "datastores": [
          {
            "entity_id": "10000:80:1910974952",
            "entity_type": "Datastore"
          },
          {
            "entity_id": "10000:80:889230871",
            "entity_type": "Datastore"
          },
          {
            "entity_id": "10000:80:1910975665",
            "entity_type": "Datastore"
          },
          {
            "entity_id": "10000:80:1910975696",
            "entity_type": "Datastore"
          },
          {
            "entity_id": "10000:80:1910974983",
            "entity_type": "Datastore"
          }
        ],
        "service_tag": "",
        "vendor_id": "host-141",
        "maintenance_mode": "NOTINMAINTENANCEMODE",
        "connection_state": "CONNECTED"
      },
      "time": 1523605419
    }
  ]
}
```


Table 6-1. Entity Type to URI

EntityType	URI
VirtualMachine	`/entities/vms`
EC2Instance	`/entities/vms`

Table 6-1. Entity Type to URI (Continued)

EntityType	URI
Host	/entities/hosts
Vnic	/entities/vnics
Vmknic	/entities/vmknics
VxlanLayer2Network	/entities/layer2-networks
VlanLayer2Network	/entities/layer2-networks
Cluster	/entities/clusters
SecurityTag	/entities/security-tags
ResourcePool	/entities/resouce-pools
NSXIPSet	/entities/ipsets
EC2IPSet	/entities/ipsets
NSXSecurityGroup	/entities/security-groups
EC2SecurityGroup	/entities/security-groups
Flow	/entities/flows
ProblemEvent	/entities/problems
NSXFirewallRule	/entities/firewall-rules
EC2SGFirewallRule	/entities/firewall-rules
NSXRedirectRules	/entities/firewall-rules
VCenterManager	/entities/vcenter-managers
NSXVManager	/entities/nsxv-managers
NSXService	/entities/services
EC2Service	/entities/services
NSXDistributedFirewall	/entities/firewalls
EC2Firewall	/entities/firewalls
NSXServiceGroup	/entities/service-groups
DistributedVirtualSwitch	/entities/distributed-virtual-switches
DistributedVirtualPortgroup	/entities/distributed-virtual-portgroups
VCDatacenter	/entities/vc-datacenters
Datastore	/entities/datastores
Folder	/entities/folders

Creating Applications and Tiers

7

vRNI provides APIs to create Applications and Tiers.

A Tier definition is based on membership criteria that can either be search based or IP address/subnet based. These applications and tiers can be used to fetch the recommended rules based on the traffic flow recorded by vRealize Network Insight from the environment.

Create an Application

Prerequisite

Ensure that you have the name of the application. Name must be unique and must not conflict with an existing application in the system.

Procedure

To create the application, make a POST request. For example:

```
Request
    POST https://vrni.example.com/api/ni/groups/applications
Request Body
    {
        "name" : "App-1",
    }
Response Body
    {
            "entity_id": "18230:561:271275765",
            "name": "App-1",
            "entity_type": "Application",
           "create_time": 1509410056733,
           "created_by": "admin@local",
           "last_modified_time": 0,
          "last_modified_by": ""
    }
```

Create a Tier Using Search Membership Criteria

Prerequisite

Ensure that you have the entity ID of the application. Name of the tier must be unique.

Procedure

To create the tier, make a POST request. For example:

```
Request
    POST https://vrni.example.com/api/ni/groups/applications/18230:561:271275765/tiers
Request Body
    {
          "name": "tier-1",
          "group_membership_criteria" : [
              {
                  "membership_type": "SearchMembershipCriteria",
                  "search_membership_criteria": {
                      "entity_type" : "VirtualMachine",
                    "filter": "security_groups.entity_id = '18230:82:604573173'"
              }
             }
          ]
    }
Response Body
        {
        "entity_id": "18230:562:1266458745",
            "name": "tier-1",
            "entity_type": "Tier",
            "group_membership_criteria": [
                {
                    "membership_type": "SearchMembershipCriteria",
                    "search_membership_criteria": {
                            "entity_type": "VirtualMachine",
                            "filter": "security_groups.entity_id = '18230:82:604573173'"
                        }
                    }
            ],
            "application": {
                "entity_id": "18230:561:271275765",
                    "entity_type": "Application"
           }
    }
```

Create a Tier Using IP Address Membership Criteria

Prerequisite

Ensure that you have the entity ID of the application. Name of the tier must be unique.

Procedure

To create the tier, make a POST request. For example:

```
Request
    POST https://vrni.example.com/api/ni/groups/applications/18230:561:271275765/tiers
Request Body
    {
          "name": "app-1-tier-2",
```

```
            "group_membership_criteria": [
                  {
                        "membership_type": "IPAddressMembershipCriteria",
                        "ip_address_membership_criteria": {
                              "ip_addresses": [
                                  "11.122.2.212",
                               "11.122.1.0/24"
                                  ]
                         }
                    }
            ]
        }
    Response Body
        {
              "name": "app-1-tier-2",
              "group_membership_criteria": [
                    {
                          "membership_type": "IPAddressMembershipCriteria",
                          "ip_address_membership_criteria": {
                                "ip_addresses": [
                                    "11.122.2.212",
                               "11.122.1.0/24"
                                  ]
                           }
                    }
              ],
        "application": {
                    "entity_id": "18230:561:271275765",
                       "entity_type": "Application"
               }
        }
```

Generating the Recommended Firewall Rules

8

vRealize Network Insight provides APIs to generate the recommended firewall rules based on the flow data.

Recommended firewall rules API provides the service to retrieve recommended rules based on flow traffic that is observed between two groups or for a single group based on all the inbound and outbound traffic for that group. If two groups are provided, both groups must be of the same type. Groups that are currently supported include Application, Tier, NSXSecurityGroup, and EC2SecurityGroup. You can provide *time_range* to determine the flow traffic that is considered for the recommended rules computation. If *ime_range* is not provided, flow traffic for the last 24 hours is considered.

Retrieve the Recommended Rules for an Application

Prerequisite

Verify that you have the entity ID of the application. Use the search service to determineRetrieve recommended the application entity ID from application name.

Procedure

1 To determine the recommended rules for the application, make a POST request with the entity ID of the application. In the following sample request, rules are computed for the application with entity ID 10000:561:1663604768.

```
Request
POST https://vrni.example.com/api/ni/micro-seg/recommended-rules

Request body:
{
  "group_1": {
      "entity": {
        "entity_type": "Application",
        "entity_id": "10000:561:1663604768"
      }
  },
  "time_range": {
        "start_time": 1508993971275,
        "end_time": 1509080371275
    }
}
```

```
Response body:
{
    "recommended_rules": [
        {
            "sources": [
                {
                    "entity_id": "10000:562:1904698621",
                    "entity_type": "Tier"
                }
            ],
            "destinations": [
                {
                    "entity_id": "10000:562:1780351215",
                    "entity_type": "Tier"
                }
            ],
            "protocols": [
                "UDP"
            ],
            "port_ranges": [
                {
                    "start": 53,
                    "end": 53
                },
                {
                    "start": 1025,
                    "end": 1025
                }
            ],
            "action": "ALLOW"
        },
        ...
    ],
    "time_range": {
        "start_time": 1508993971275,
        "end_time": 1509080371275
    }
}
```

Retrieve the Recommended Rules Between Two Tiers

Prerequisite

Ensure that you have the entity ID of Tier1 and Tier2.

Procedure

1 To determine the recommended rules, make a POST request with the entiyID of the tiers. In the following sample request, rules are computed for the tier with entity ID 10000:562:190469862 and tier with entity ID 10000:562:178035121.

```
Request
POST https://vrni.example.com/api/ni/micro-seg/recommended-rules

Request body:
{
  "group_1": {
      "entity": {
        "entity_type": "Tier",
            "entity_id": "10000:562:1904698621"
      }
  },
  "group_2": {
      "entity": {
        "entity_type": "Tier",
            "entity_id": "10000:562:1780351215"
      }
  }
}

Response body:
{
    "recommended_rules": [
        {
            "sources": [
                {
                    "entity_id": "10000:562:1904698621",
                    "entity_type": "Tier"
                }
            ],
            "destinations": [
                {
                    "entity_id": "10000:562:1780351215",
                    "entity_type": "Tier"
                }
            ],
            "protocols": [
                "UDP"
            ],
            "port_ranges": [
                {
                    "start": 53,
                    "end": 53
                },
                {
                    "start": 1025,
                    "end": 1025
                }
            ],
```

```
            "action": "ALLOW"
        }
    ],
    "time_range": {
        "start_time": 1508996919391,
        "end_time": 1509083319391
    }
}
```

Export the Recommended Rules for an Application in the NSX Compatible Format

You can also export the recommended firewall rules and security groups in NSX compatible format as a ZIP file using this API.

Prerequisite

Ensure that you have the entity ID of the application. Use the search service to fetch the application ID from the name.

Procedure

1 To determine the recommended rules for the application, make a POST request using the application entity ID.

For example:

```
Request
POST https://vrni.example.com/api/ni/micro-seg/recommended-rule/nsx
Request Header
Accept-Type: "application/octet-stream"
Request Body:
{
  "group_1": {
      "entity": {
        "entity_type": "Application",
        "entity_id": "10000:561:1663604768"
      }
  },
  "time_range": {
        "start_time": 1508993971275,
        "end_time": 1509080371275
    }
}

Response
```

Zip File containing the NSX artifacts.

Exporting Recommended Rules Between Two Tiers in NSX Compatible Format

Prerequisite

Ensure that you have the entity ID of Tier1 and Tier2.

Procedure

1 To determine the recommended rules, make a POST request.

For example:

```
Request
POST https://vrni.example.com/api/ni/micro-seg/recommended-rule/nsx
Request Header
Accept-Type: "application/octet-stream"
Request Body:
{
  "group_1": {
      "entity": {
        "entity_type": "Tier",
            "entity_id": "10000:562:1904698621"
      }
  },
  "group_2": {
      "entity": {
        "entity_type": "Tier",
            "entity_id": "10000:562:1780351215"
      }
  }
}
Response
```

Zip File containing the NSX artifacts.

Fetching Metrics

9

Metrics can be fetched for an entity based using this API. Metrics schema APIs give the available metrics and their intervals for an entity type.

Retrieve Metrics Schema for an Entity Type

Prerequisite

Verify that you have the entity type for which metrics are required.

Procedure

1 To determine the metrics info for an entity type, make a GET request with the entity type of the application as a path parameter. In the following sample request, metrics info for a FLOW entity type is retrieved.

```
Request
GET https://vrni.example.com/api/ni/schema/Flow/metrics

Response body:
{
    "results": [
        {
            "metric": "flow.dstBytes.delta.summation.bytes",
            "display_name": "Destination Bytes",
            "intervals": [
                1800,
                7200,
                28800
            ],
            "description": "Total bytes sent by the server to client",
            "unit": "NoUnit"
        },
        {
            "metric": "flow.srcBytes.delta.summation.bytes",
            "display_name": "Source Bytes",
            "intervals": [
                1800,
                7200,
                28800
            ],
```

```
            "description": "Total bytes sent by the client to server",
            "unit": "NoUnit"
        },
        {
            "metric": "flow.totalBytes.delta.summation.bytes",
            "display_name": "Bytes",
            "intervals": [
                1800,
                7200,
                28800
            ],
            "description": "Total bytes transferred in both directions",
            "unit": "NoUnit"
        },
        {
            "metric": "flow.totalBytesRate.rate.average.bitsPerSecond",
            "display_name": "Bytes Rate",
            "intervals": [
                1800,
                7200,
                28800
            ],
            "description": "Average Bits/sec rate of total traffic in both directions",
            "unit": "NoUnit"
        },
        {
            "metric": "flow.totalPackets.delta.summation.number",
            "display_name": "Packets",
            "intervals": [
                1800,
                7200,
                28800
            ],
            "description": "Total packets transferred in both directions",
            "unit": "NoUnit"
        },
        {
            "metric": "flow.allowedSessionCount.delta.summation.number",
            "display_name": "Session Count",
            "intervals": [
                1800,
                7200,
                28800
            ],
            "description": "Total number of allowed sessions",
            "unit": "NoUnit"
        }
    ]
}
```


Retrieve Metrics Points for an Entity

Prerequisite

Verify that you have the entity id, metrics name, interval, start, and end time.

Procedure

1 To fetch metrics points for an entity id and metric for a given time interval , make a GET request as shown in the following example . Maximum number of metrics point returned by API is 300 and a 400 response is returned in case the interval and time period combination have more than 300 metrics points. The client can break the time period to multiple batches to get all the metrics points.

```
Request
GET https:/vrni.example.com/api/ni/metrics?entity_id=12347:515:1787493997
&metric=flow.srcBytes.delta.summation.bytes&interval=1800&start=1523351589&end=1523437989

Response body:
{
    "metric": "flow.srcBytes.delta.summation.bytes",
    "display_name": "Source Bytes",
    "interval": 1800,
    "unit": "NoUnit",
    "pointlist": [
        [
            1523352600,
            7141
        ],
        [
            1523354400,
            6898
        ],
        [
            1523356200,
            7370
        ],
        [
            1523358000,
            6898
        ]
    ],
    "start": 1523351589,
    "end": 1523437989
}
```

Get Proxy Node Details

10

These APIs are used to fetch all proxy nodes details. Proxy node id is required for adding data source.

Get Proxy Nodes List

Procedure

1. To fetch a list of proxy nodes, make a GET request as shown below.

```
GET https:/vrni.example.com/api/ni/infra/nodes
{
  "results": [
    {
       "id": "10000:901:1575899868",
       "entity_type": "NODE"
    }
    ],
  "total_count": 1
}
```

Get Proxy Node Details

Procedure

1. To fetch proxy nodes details, make a GET request as shown below.

```
GET https:/vrni.example.com/api/ni/infra/nodes/10000:901:1575899868
Response
{
   "id": "10000:901:1575899868",
   "entity_type": "NODE",
   "node_type": "PROXY_VM",
   "node_id": "IPE97DK",
   "ip_address": "10.153.190.94"
}
```

Get Version Info

11

This API returns API version.

Get API Version Info

```
Request
GET https://vrni.example.com/api/ni/info/version
Response
{ "api_version": "1.1.0" }
```