ivanti

Data Center Discovery 2020.2

REST API QUICK START GUIDE

Rev 10/20

Contents

About this guide

This is a quick start guide to Ivanti's Data Center Discovery REST API. It covers the basics of how to access the API, how to authenticate, and how to navigate between resources. You can find more detail about the resources supported and the data returned by each resource by checking the Swagger documentation.

Accessing the API

The REST API is accessible using the following URLs:

```
http://<server_ip_hostname>/api/<api_version>/<resource_name>

http://<server_ip_hostname>/api/<resource_name>
```

- **<server_ip_hostname>** is the hostname of the server where the REST API is installed.
- **<api_version>** is the version of the API (e.g., v1).
- **<resource_name>** is the resource being requested (e.g., /devices). To check which other resources are available and to find out more about them, refer to the Swagger docs.

Authentication

The REST API uses basic authentication as described by the standard RFC2617. To execute basic authentication, an Authentication header must be added and sent with every request:

HTTP headers

Authorization	Basic YWRtaW46cGFzc3dvcmQ=

Where **Authorization** is the header name and the header value is in the form of:

```
Basic  <encoded_credential>
```

The encoded credential is the concatenation of the username, a colon, and the password, which is then encoded in base64:

```
Base64 of:   <username>:<password>
```

The REST API has a Dry Run (test) mode that you can use to check the credential. This is accessible by doing a request to the root address of the API:

```
http://<server_ip_hostname>/api/v1
```

If the credential used is valid, a **200 – OK** is returned.

Making requests

This section covers some possible requests and the navigation between related resources.

The most typical request that can be performed is a request for devices. To request all devices, the following resource can be invoked:

```
http://<server_ip_hostname>/api/v1/devices
```

Using Curl, together with the Authorization header:

```
curl -H 'Authorization: Basic YWRtaW46cGFzc3dvcmQ='
http://<server_ip_hostname>/api/v1/devices'
```

When querying a resource that returns a collection such as **/devices** or **/applications**, by default a minimal view of the objects is returned. This minimal view contains a limited number of properties, enough to identify each one:


```
[
 {
  "self": "http://localhost/API/v1/devices/C34474BB-0A31-49EC-BB2E-922777F55960",
  "device_id": "C34474BB-0A31-49EC-BB2E-922777F55960",
  "host_name": "VM-EXCHANGE-213",
  "serial_number": "VMware-42 37 0e b4 28 a1 6c b4-12 d5 42 a2 37 dc ea 5d",
  "manufacturer": "VMware, Inc.",
  "model": "VMware Virtual Platform",
  "last_scan": "",
  "device_type_hint": "Windows Server Operating Systems",
  "qualified_name": [
   {
    "name": "vm-exchange-213.qalab.local",
    "name_type": "FQDN"
   }
  ]
 }
 … additional items omitted …
]
```


To drill down into a single object and return all its properties, a new request should be made to the URL previously returned in the **self** field:


```
{
  "self":"http://localhost/API/v1/devices/C34474BB-0A31-49EC-BB2E-922777F55960",
  "device_id":"C34474BB-0A31-49EC-BB2E-922777F55960",
  "host_name":"VM-EXCHANGE-213",
  "serial_number":"VMware-42 37 0e b4 28 a1 6c b4-12 d5 42 a2 37 dc ea 5d",
  "manufacturer":"VMware, Inc.",
  "model":"VMware Virtual Platform",
  "last_scan":"",
```

```
  "device_type_hint":"",
  "qualified_name":[
    {
      "name":"vm-exchange-213.qalab.local",
      "name_type":"FQDN"
    }
  ],
  "is_hypervisor": false,
  "users": "http://localhost/API/v1//devices/C34474BB-0A31-49EC-BB2E-922777F55960/users",
  "total_memory_mb": 6143,
  "cpu_count": 2,
  "core_count": 2,
  "is_virtual": true,
  "operating_system": {
    "name": "Microsoft Windows Server 2012 R2 Standard",
    "version": "",
    "edition": "",
    "build_number": "6.3.9600",
    "service_pack": "",
    "product": {
      "name": "Windows Server Operating Systems",
      "vendor": "Microsoft",
      "description": "Queries Active Directory Servers for information using the LDAP protocol"
    }
  },
  "applications": "http://localhost/API/v1/devices/C34474BB-0A31-49EC-BB2E-
922777F55960/applications",
  "installed_software": "http://localhost/API/v1//devices/C34474BB-0A31-49EC-BB2E-
922777F55960/installed_software",
  "bios": {
    "name": "PhoenixBIOS 4.0 Release 6.0",
    "manufacturer": "Phoenix Technologies LTD",
    "version": "INTEL  - 6040000"
  },
  "os_install_date": "2017-10-11 11:32:53",
  "components": "http://localhost/API/v1//devices/C34474BB-0A31-49EC-BB2E-922777F55960/components",
  "virtual_name": "vm-exchange-2k13",
  "virtual_host": {
  },
  "virtualization_platform": "VMware",
  "power_state": "poweredOn",
  "evc_enabled": "false",
  "state": "",
  "uptime": "2018-10-22 14:25:53",
  "vmotion_enabled": "",
  "partition_type": "",
  "partition_mode": "",
  "entitled_capacity": "2.00000",
  "online_virtual_cpu": "2",
  "cpu_pool": {
```

```
    "shared_pool_id": "",
    "active_cpu_count": null
  },
  "max_capacity": "",
  "cpu": [
    {
      "index": 0,
      "manufacturer": "GenuineIntel",
      "cpu_model": "Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz",
      "core_count":  1,
      "threads_per_core": 1,
      "status": "",
      "cpu_type": "Intel64 Family 6 Model 63 Stepping 0",
      "cpu_speed": 3.196,
      "is_hyperthreaded": false
    },
    {
      "index": 1,
      "manufacturer": "GenuineIntel",
      "cpu_model": "Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz",
      "core_count": 1,
      "threads_per_core": 1,
      "status": "",
      "cpu_type": "Intel64 Family 6 Model 63 Stepping 0",
      "cpu_speed": 3.196,
      "is_hyperthreaded": false
    }
  ],
  "ip_address":[
    {
      "ip_address": "192.168.0.119",
      "mac_address": "00:50:56:B7:33:0B",
      "is_verified": true,
      "is_virtual": true,
      "is_dhcp_enabled": true,
      "dhcp_lease_obtained_gmt": null,
      "dhcp_lease_expired_gmt": null,
      "dhcp_server": null,
      "network_interface": "Ethernet0"
    },
   ],
  "logical_disk":[ … collapsed … ],
  "process":[ … collapsed … ],
  "windows_service":[ … collapsed …]
}
```


Following the same logic, to return all applications running on this device, it's a matter of requesting the resource returned in the **applications** field for the returned device object.

Column sets

Column set parameters are the means to control the number of columns returned with a request. Currently supported **col_set** values are **default** and **all**.

As highlighted previously, the **/devices** collection returns a minimal (default) view of each device. However, it might be required to get the full device information by making a single request. This is achieved by providing a parameter to request all columns associated with a device.

To do this, use the **col_set** URL parameter and set its value to **all** as follows:

http:// <server_ip_hostname>/api/v1/devices/?col_set=all

This will return the list of devices, listing each with all of their information.

Pagination of collections

It's possible to get limited or more extensive batches of collections. This is done using the **offset** and **fetch_size** URL parameters. For example:

http:// <server_ip_hostname>/api/v1/devices/?offset=50&fetch_size=25

This request returns the device list starting on the 50th device (inclusive) and returns a total of 25 devices.

When these are not specified, **offset** defaults to 0 and **fetch_size** defaults to 200 or 20 (depending on the **col_set** defined).

Navigating between paginated collections

Collections are paginated according to the number of objects returned. To facilitate the navigation between the set of results, a URL pointing to the next page of results is included in the Response header:

```
"link": "<http://localhost/API/v1/devices/?offset=12&fetch_size=2&col_set=all>; rel="next",
```

"Link" is always returned when a collection is returned, independent of the existence of more results in that collection.

Filtering results

It's possible to filter the results according to some of their properties.

To send a request with filtering parameters, the request type must be set as **POST** and the parameters sent as **x-www-form-urlencoded** data.

Wildcards are supported by appending and/or prepending **%** to the filtering parameter.

Deleting a target set configuration

All target configurations can be cleared down by invoking the following endpoint resource with the **DELETE** method:

```
http://<server_ip_hostname>/api/v1/targets
```

This will delete all target sets and any data gathered from scanning them. It will not remove credentials, connection settings, scan windows, or any other location configurations.

Because a target set clear down could take some time (depending on the size of the estate), requests to this endpoint will confirm the receipt of the request and issue a Request ID in the response.

```
{
 "message": "Request accepted and queued, RequestID is 30",
 "request_id": "30"
}
```

The Request ID can be used to check on the status of the clear-down task by sending it to the following endpoint resource with the **GET** method:

```
http:// <server_ip_hostname>/api/v1/targets/deletestatus/{request_id}
```

One of the following results will be returned:

Status	Description
Complete	The procedure completed successfully.
Pending	The procedure is in progress.
Invalid Request ID	The Request ID provided is incorrect.

Performing a target set configuration

You can perform a target set configuration by invoking the following endpoint resource with the **POST** method and providing the target configuration in the body of the request:

```
http://<server_ip_hostname>/api/v1/targets
```

The required target configuration is a correctly formatted JSON string for the sets of targets to be created. The formatting is shown below.

The configuration of the target set could take some time, depending on the size of the estate to be created; therefore, requests to this endpoint are queued. The response to this request contains a Request ID:

```
{
 "message": "Request accepted and queued, RequestID is 31",
 "request_id": "31"
}
```

You can use this Request ID to check on the status of your target set configuration by using the following endpoint resource with the **GET** method:

```
http://<server_ip_hostname>/api/v1/targets/poststatus/{request_id}
```

This response will contain a message indicating the status of request:

```
{
 "message": "Completed",
 "details": "",
 "request_id": "1"
}
```

The following table describes the possible status responses:

Status	Description
Queued	The REST API request is queued.
Processing	The REST API request is being processed.
Completed	The REST API request has been successfully processed.
Error	There was a fatal error processing the REST API request.
Invalid Request ID	The Request ID provided is invalid.

The response also contains a details field. If the status is **Error**, the details field will provide more information regarding the error. You may also receive details when the request status is **Completed**. This will contain messages/warnings regarding the operation that you were attempting to perform.

Target configuration JSON

The target configuration should be supplied as follows:


```
{
 "Name":"LOCATION TARGET",
 "Version":"1.3",
 "Locations":
  [
   {
    "LocationPath":"QALab|Main_Lab",
    "Target":"Device",
    "Type":"Hostname",
    "Name":"Hostname Target",
    "InstanceName":"",
    "Guid":"2b985b3f-8584-4c09-bcc7-0284b644c05e",
    "Hostname":"VM-IBM1-2K8",
    "StartIP":"",
    "EndIP":"",
    "SubnetMask":"0",
    "Port":"",
    "Exclusion":""
   },
   {
    "LocationPath":"QALab|Main_Lab",
    "Target":"Device",
    "Type":"Range",
    "Name":"Main lab Dot 4 range 1",
    "InstanceName":"",
    "Guid":"8651e81f-c92f-49cd-87df-7a87577685a4",
    "Hostname":"",
```

```
      "StartIP":"192.168.4.0",
      "EndIP":"192.168.4.108",
      "SubnetMask":"",
      "Port":"",
      "Exclusion":"False"
    }
   ]
}
```


The **Name** must be **LOCATION TARGET** and the **Version** must be **1.3**. You then have an array of locations. Specify the relevant fields for each location as required.

Note that this is displayed here in a format to make it easier to read; since the content is JSON (that disregards white space), the actual parameter string does not need to contain the line breaks:


```
{"Name":"LOCATION TARGET","Version":"1.3","Locations":[{"LocationPath":"QALab|Main…
```


Applications

The Applications resource supports filtering by Vendor, Name, and Description. As an example, the following would return only applications from Oracle:

Form data for x-www-form-urlencoded parameters

vendor	Oracle

Using Curl:

```
curl -H 'Authorization: Basic YWRtaW46cGFzc3dvcmQ=' -d 'vendor=oracle' -X POST
'http://<server_ip_hostname>/api/v1/applications'
```

Response codes

The following are the HTTP response codes that can be returned by the REST API:

HTTP Code	Description
200	OK – The HTTP request was successful.
401	Unauthorized – The authentication failed. Double-check the credentials used and the format.
404	Not found – The requested resource does not exist.
500	General server error.