



# Data Center Discovery 2019.2

---

REST API QUICK START GUIDE

**Contents**

About this guide ..... 4

    Accessing the API ..... 4

    Authentication ..... 4

    Making requests..... 5

        Column sets ..... 6

        Pagination of collections..... 6

        Navigating between paginated collections ..... 7

    Filtering results..... 7

    Performing a target set configuration..... 7

        Target configuration JSon ..... 8

    Deleting a target set configuration ..... 9

    Applications..... 10

    Response codes ..... 10

**Copyright notice**

This document is provided strictly as a guide. No guarantees can be provided or expected. This document contains the confidential information and/or proprietary property of Ivanti, Inc. and its affiliates (referred to collectively as “Ivanti”) and may not be disclosed or copied without prior written consent of Ivanti.

Ivanti retains the right to make changes to this document or related product specifications and descriptions, at any time, without notice. Ivanti makes no warranty for the use of this document and assumes no responsibility for any errors that can appear in the document nor does it make a commitment to update the information contained herein. For the most current product information, please visit [www.ivanti.com](http://www.ivanti.com).

Copyright © 2019, Ivanti. All rights reserved.

Ivanti and its logos are registered trademarks or trademarks of Ivanti, Inc. and its affiliates in the United States and/or other countries. Other brands and names may be claimed as the property of others.

Protected by patents, see <http://www.ivanti.com/patents>

Rev 11/19

## 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 URL:

```
http://<server_ip_hostname>/api/<api_version>/<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 do basic authentication, a header is added and sent with every request:

HTTP headers

<u>Authorization</u>	<u>Basic YWRtaW46cGFzc3dvcmQ=</u>
----------------------	-----------------------------------

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>
```

For example, the default Data Center Discovery credential (admin/password) becomes `admin:password`, which after encoding becomes `YWRtaW46cGFzc3dvcmQ=`, and after prepending the `Basic` string, becomes `Basic YWRtaW46cGFzc3dvcmQ=`.

Base64 encoding is supported by default on UNIX systems or via Cygwin on Windows systems. Be careful not to encode hidden characters like newlines or tabs in the string.

**Unix:**

```
$ echo -n admin:password | base64
```

**Windows (via Cygwin):**

```
$ echo -n "admin:password" | base64.exe
```

The REST API has a Dry Run 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 exemplifies 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://vm-devse-pc0/api/v1/devices
```

Using Curl, together with the Authorization header:

```
curl -H 'Authorization: Basic YWRtaW46cGFzc3dvcmQ=' 'http://vm-devse-pc0/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:

```
- 4: {
  "device_id": "d4281ee6-b0aa-46e7-9a6b-5390353ecf59",
  "host_name": "Solarisv490",
  "serial_number": "84ad4eee",
  "manufacturer": "Oracle Corporation sun4u",
  "model": "Oracle Corporation sun4u Sun Fire V490",
  "self": "http://vm-devse-pc0/API/v1/devices/D4281EE6-B0AA-46E7-9A6B-5390353ECF59"
}
```

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:

```
{
  "device_id": "d4281ee6-b0aa-46e7-9a6b-5390353ecf59",
  "host_name": "Solarisv490",
  "total_memory_mb": 8192,
  "serial_number": "84ad4eee",
  "manufacturer": "Oracle Corporation sun4u",
  "model": "Oracle Corporation sun4u Sun Fire V490",
  "cpu_count": 2,
  "core_count": 4,
  "is_virtual": false,
  - "operating_system": { ... }
  - "cpu": Array[2] ...
  - "ip_address": Array[3] ...
  "applications": "http://vm-devse-pc0/API/v1/devices/D4281EE6-B0AA-46E7-9A6B-5390353ECF59/applications",
  "installed_software": "http://vm-devse-pc0/API/v1/devices/D4281EE6-B0AA-46E7-9A6B-5390353ECF59/installed\_software",
  "self": "http://vm-devse-pc0/API/v1/devices/D4281EE6-B0AA-46E7-9A6B-5390353ECF59"
}
```

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

As highlighted previously, the `/devices` collection returns a minimal view of each device. However, it might be required to get the full device information in one go.

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

```
http://vm-devse-pc0/api/v1/devices/?col_set=all
```

This will return the list of devices, listing each with all of their information. It's worth noting that setting this option limits the number of returned results to 20.

## 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://vm-devse-pc0/api/v1/devices/?offset=50&fetch_size=25
```

This request returns the device list starting on the 50<sup>th</sup> 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://vm-devse-pc0/API/v1/devices/?offset=162&fetch\\_size=0&col\\_set=default](http://vm-devse-pc0/API/v1/devices/?offset=162&fetch_size=0&col_set=default)>; 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, which is especially useful if the API is used in front-end searching or filtering.

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.

## 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://vm-devse-pc0/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://vm-devse-pc0/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; the actual parameter string does not need to contain the line breaks:

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

## Deleting a target set configuration

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

```
http://vm-devse-pc0/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 clearing down a target set 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. 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://vm-devse-pc0/api/v1/targets/deletestatus/{request_id}
```

One of the following results will be shown:

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

## 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://vm-devse-pc0/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.