Overview

The following topics will help you learn how to connect to Veeam Service Provider Console REST API and authorize your access. You will also learn what method-implemented operations you can perform with Veeam Service Provider Console entities, how to control and sort the data that Veeam Service Provider Console REST API returns, and monitor request statistics.

Resource URLs

Each Veeam Service Provider Console REST API-exposed entity has a resource representing it. To get a resource representation, request its URL. All Veeam Service Provider Console REST API URLs have the following base:

```
https://<hostname>:<port>/api/v3
```

where:

- `<hostname>` is the DNS name or IP address of the machine on which Veeam Service Provider Console Web UI is installed.
- `<port>` is the REST API v3 port number you specified during Veeam Service Provider Console Web UI installation.

  The default port number is `1280`.

To get a collection of the Veeam Service Provider Console REST API resources, request a base URL followed by resource collection path. For example:

```
GET https://localhost:1280/api/v3/infrastructure/backupServers
```

To get a representation of an individual resource, enter a resource identifier after a resource collection path. For example:

```
GET https://localhost:1280/api/v3/infrastructure/backupServers/c3517470-112c-11ea-aaef-0800200c9a66
```


The resource identifier is represented by the `instanceUid` properties of the resource from the collection.


```
{
  "meta": {
    "pagingInfo": {
      "total": 3,
      "count": 3,
      "skip": 0
    }
  },
  "data": [
    {
      "instanceUid": "c3517470-112c-11ea-aaef-0800200c9a66",
      "name": "bckp2",
      "locationUid": "140b6824-5b8c-437f-a2b9-1bf6dc4a9790",
      "managementAgentUid": "a07794fe-3585-4388-9c75-2f960c19197d",
      "version": "11.0.0.837",
      "displayVersion": "11.0.0.837",
      "installationUid": "9b45a1fa-8bfb-4f23-890d-aa56408faf73",
      "backupServerRoleType": "cloudConnect",
      "status": "healthy"
    },
    {
      "instanceUid": "38110067-0c67-462b-9dfa-656bc08cd3b3",
      "name": "bckp3",
      "locationUid": "140b6824-5b8c-437f-a2b9-1bf6dc4a9790",
      "managementAgentUid": "5c50d211-791b-4940-b547-db740e96b7a0",
      "version": "11.0.0.837",
      "displayVersion": "11.0.0.837",
      "installationUid": "889e15e0-bfb7-4804-ab0e-5a97279d8fde"
```

```
      "installationUid": "[illegible]",
      "backupServerRoleType": "cloudConnect",
      "status": "healthy"
    },
    {
      "instanceUid": "450e2a1b-f4a7-4ebc-b246-c32413fa15b2",
      "name": "bckp6",
      "locationUid": "3f5938c4-12c5-45a4-a1a5-c158d5848238",
      "managementAgentUid": "eb4c38b4-8569-4d51-91d2-00ba90c06ccc",
      "version": "11.0.0.837",
      "displayVersion": "11.0.0.837",
      "installationUid": "49ce8088-aba6-403c-bd7d-1d734472e50a",
      "backupServerRoleType": "client",
      "status": "healthy"
    }
  ]
}
```


Authorization and Security

To start working with the Veeam Service Provider Console REST API, clients must first authenticate themselves and receive authorization grant by obtaining an access token and a refresh token.

- **Access token** is a string that represents authorization issued to the client and must be used in all requests. By default, access token expires after 1 hour.
- **Refresh token** is a string that represents authorization granted to the client and can be used to obtain a new access token when the current access token expires.

You authorize to Veeam Service Provider Console REST API using the following methods:

- OAuth 2.0 Authentication.

- API Key-Based Authorization.

OAuth 2.0 Authentication

This authentication type is based on the OAuth 2.0 Authorization Framework.

To obtain a pair of tokens, send the HTTP POST request to the `/token` path.

A successfully completed operation returns the *200 OK* response code and an access and a refresh token in the response body. The client inserts the access token in headers of further requests to the Veeam Service Provider Console REST API. The refresh token must be saved locally.

> **TIP**
>
> To learn how to authorize your access using an application, you can review the Example Requests and Responses section. Alternatively, you can use Swagger UI.

Using Refresh Token

To obtain a new pair of tokens in case the access token expires or becomes invalid, send the HTTP POST request with the refresh token in the request body to the `/token` path. A successfully completed operation returns the *200 OK* response code and a new pair of tokens in the response body.

Token Invalidation

An access token expires in 1 hour which means that the access to resources is disabled automatically. If an access token must be revoked immediately, send the HTTP DELETE request to the `/users/{userId}/tokens` path where `{userId}` is a UID assigned to a user.

To revoke a token of a specific user identity, specify ID assigned to this user identity in the `userLoginID` query parameter.

A successfully completed operation returns the *200 OK* response code and the `true` value in the response body.

NOTE

To learn how to authorize your access using an application, you can review the Example Requests and Responses section. Alternatively, you can use Swagger UI.

Example Requests and Responses

The following example illustrates how you communicate with the server using requests and responses.

1. To obtain an access and a refresh token, send the HTTP POST request with the `application/x-www-form-urlencoded` content type to the `/token` path.

   In the body of the request, specify the following parameters:

   - grant_type — the authorization process requires that the password value is specified for this parameter.
   - username and password — credentials used to access the server; in this example, vspc\administrator and Password1 are used.

```
Request:
POST https://vspc:1280/api/v3/token

Request Header:
Content-Type: application/x-www-form-urlencoded
Authorization: Bearer

Request Body:
grant_type : password
username : vspc\administrator
password : Password1
```

The server sends a response in the following format.

```
Response Code:
200 OK

Response Body:
{
  "access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOiIxNTgzMDA4MTc4IiwiZXhwIjoiMTA4MzAxMTc3OCIsInN1YiI6I
UDRj79NqiZQ-
FZI1Jyre4HI3r4undfP9Y3BxSxswLD61XNGao1VWIDf2UdpTxIekimeaPS12Km0YY2prWp5jkvMHe5IR_JQWi6D-
DeYf5Smdcn4fVNpsb327qdONf1Vp2pgkLuEZim33Two4r8cDXj3q6h2NCOnxf1wD3Kv5fvLxT33G6Ia37kiCjdwKI2MWlyppaoL
2WSdfN3QzlA",
  "token_type": "bearer",
  "refresh_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOiIxNTgzMDA4MTc4IiwiZXhwIjoiMTU4MzE4MDk3OCIsInN1YiI6I
J5uKX7-dYFCds9mdPBxC3-_glCoVciPwJZR82MLYP5lZe5Rq56KbzjmEA3BlS5wx0j7jK75ZNFgM-
Y4gEZSYmlAxArrM7DvK1CPoGDib_XAWqNw2mBNUZloGe0yTh8FvVQlw2Hb8NddMmfJdCi5JRrSguRgX-
Z1kojI8Zx7HVWAanG8woI_YfvBamN4_NVJRbQR10iLbYNK5_kclK9YmJC4rmC0RgO2FSXa-
0gMldmAQ_7iDERBDdmHBpWRJTwZblYK40vSJYNz27cq-G9BAyQR-UqwbAYzTBenn7S99FwGzkA_W1NeyBBq5Va-
nFZL7rY_h6TrfcKGRSrvA"
  "mfa_token": null,
  "encrypted_code": null,
  "expires_in": 3600,
}
```

After you authorize you can authenticate as another user. For details, see User Impersonation.

2. To refresh a pair of tokens, send the HTTP POST request with the `application/x-www-form-urlencoded` content type to the `/token` path.

   In the body of the request, specify the following values for the parameters:

   - `grant_type` – to refresh the token, it is required that the `refresh_token` value is specified for this parameter

- `grant_type` — to refresh the token, it is required that the `refresh_token` value is specified for this parameter.
- `refresh_token` — the previously saved refresh token.

```
Request:
POST https://vspc:1280/api/v3/token

Request Header:
Content-Type: application/x-www-form-urlencoded
Authorization: Bearer

Request Body:
grant_type : refresh_token
refresh_token :
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOiIxNTgzMDA4MTc4IiwiZXhwIjoiMTU4MzE4MDk3OCIsInN1YiI6Ij
J5uKX7-dYFCds9mdPBxC3-_glCoVciPwJZR82MLYP5lZe5Rq56KbzjmEA3BlS5wx0j7jK75ZNFgM-
Y4gEZSYmlAxArrM7DvK1CPoGDib_XAWqNw2mBNUZloGe0yTh8FvVQlw2Hb8NddMmfJdCi5JRrSguRgX-
Z1kojI8Zx7HVWAanG8woI_YfvBamN4_NVJRbQR10iLbYNK5_kclK9YmJC4rmC0RgO2FSXa-
0gMldmAQ_7iDERBDdmHBpWRJTwZblYK40vSJYNz27cq-G9BAyQR-UqwbAYzTBenn7S99FwGzkA_W1NeyBBq5Va-
nFZL7rY_h6TrfcKGRSrvA
```

User Impersonation

In Veeam Service Provider Console, you can use OAuth 2.0 protocol to impersonate another user. It can be useful if you want to perform REST API operations on behalf of a user of a managed company or reseller.

Required Privileges

To perform this task, a user must have one of the following roles assigned: Portal Administrator, Location Administrator, Site Administrator, Portal Operator, Service Provider Global Administrator, Service Provider Administrator, Company Owner, Service Provider Operator, Company Administrator.

Note that to impersonate another user you are required to have the same permissions as for editing that user.

Note that to impersonate another user you are required to have the same permissions as for editing that user.

Impersonating Another User

To authenticate as another user:

1. Authorize as described in the OAuth 2.0 Authentication section.
2. Send the HTTP POST request with the `application/x-www-form-urlencoded` content type to the `/token` path.

   In the body of the request, specify the following values for the parameters:

   - `grant_type` — to authenticate as another user, it is required that the `as` value is specified for this parameter.
   - `refresh_token` — the previously saved refresh token.
   - `userUid` — UID assigned to a user that you want to impersonate.

```
Request:
POST https://localhost:1280/api/v3/token

Request Header:
Content-Type: application/x-www-form-urlencoded
Authorization: Bearer

Request Body:
grant_type : as
refresh_token :
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOiIxNTgzMDA4MTc4IiwiZXhwIjoiMTU4MzE4MDk3OCIsInN1YiI6Ij
J5uKX7-dYFCds9mdPBxC3-_glCoVciPwJZR82MLYP5lZe5Rq56KbzjmEA3BlS5wx0j7jK75ZNFgM-
Y4gEZSYmlAxArrM7DvK1CPoGDib_XAWqNw2mBNUZloGe0yTh8FvVQlw2Hb8NddMmfJdCi5JRrSguRgX-
Z1kojI8Zx7HVWAanG8woI_YfvBamN4_NVJRbQR10iLbYNK5_kclK9YmJC4rmC0RgO2FSXa-
0gMldmAQ_7iDERBDdmHBpWRJTwZblYK40vSJYNz27cq-G9BAyQR-UqwbAYzTBenn7S99FwGzkA_W1NeyBBq5Va-
nFZL7rY_h6TrfcKGRSrvA
userUid : b3ffc78b-7982-44f3-ae66-1384415f2bea
```

The server sends a response in the following format.

```
Response Code:
200 OK

Response Body:
{
  "access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOiIxNTgzMDA4MTc4IiwiZXhwIjoiMTA4MzAxMTc3OCIsInN1YiI6I
UDRj79NqiZQ-
FZI1Jyre4HI3r4undfP9Y3BxSxswLD61XNGao1VWIDf2UdpTxIekimeaPS12Km0YY2prWp5jkvMHe5IR_JQWi6D-
DeYf5Smdcn4fVNpsb327qdONf1Vp2pgkLuEZim33Two4r8cDXj3q6h2NCOnxf1wD3Kv5fvLxT33G6Ia37kiCjdwKI2MWlyppaoL
2WSdfN3QzlA",
  "token_type": "bearer",
  "refresh_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOiIxNTgzMDA4MTc4IiwiZXhwIjoiMTU4MzE4MDk3OCIsInN1YiI6I
J5uKX7-dYFCds9mdPBxC3-_glCoVciPwJZR82MLYP5lZe5Rq56KbzjmEA3BlS5wx0j7jK75ZNFgM-
Y4gEZSYmlAxArrM7DvK1CPoGDib_XAWqNw2mBNUZloGe0yTh8FvVQlw2Hb8NddMmfJdCi5JRrSguRgX-
Z1kojI8Zx7HVWAanG8woI_YfvBamN4_NVJRbQR10iLbYNK5_kclK9YmJC4rmC0RgO2FSXa-
0gMldmAQ_7iDERBDdmHBpWRJTwZblYK40vSJYNz27cq-G9BAyQR-UqwbAYzTBenn7S99FwGzkA_W1NeyBBq5Va-
nFZL7rY_h6TrfcKGRSrvA"
  "mfa_token": null,
  "encrypted_code": null,
  "expires_in": 3600,
}
```

API Key-Based Authorization

Using an API key allows a user to receive a permanent authorization grant. Unlike access and refresh tokens that expire after a specific period of time, an

API key is active until the associated user identity is disabled or deleted.

To configure API key-based authentication:

1. Log in using one of the methods described in the OAuth 2.0 Authentication section.
2. Send the HTTP POST request to the `/users/{userUid}/logins/apikey` path, where `userUid` is the UID of your current user account. Additionally you must provide the following query string parameters:
   - `description` — contains information about the API key issuance.
   - `scopes` — defines array of services to which the API key grants access. You can specify the following values:
     - `ui` — allows a user to send UI requests.
     - `rest` — allows a user to send REST API requests.
     - `integration` — allows a user to configure integrations.

   If you want a user to have access only to GET operations, provide the `true` value for a `isReadAccessOnly` query parameter. The response body will contain a resource representation of a new user identity with the API key in the parameters property.

```
Request:
POST https://vspc:1280/api/v3/users/955b0b80-03b2-49fd-bf38-14785ecff0a8/logins/apikey?
description=Portal Operator API Key&scopes=["integration","rest"]&isReadAccessOnly=true

Request Headers:
Content-type: application/json
Authorization: Bearer <Access-Token>

Response Code:
200 OK

Response Body:
{
```

```
  "data": {
    "id": 12,
    "userUid": "955b0b80-03b2-49fd-bf38-14785ecff0a8",
    "userName": "alpha_admin",
    "companyId": "3b139961-6b11-4fd8-9e87-c2be1b0e9ca8",
    "companyName": "Alpha",
    "identityProviderName": "AK",
    "description": "Portal Operator API Key",
    "isReadAccessOnly": true,
    "scopes": [
      "integration",
      "rest"
    ],
    "status": "enabled",
    "parameters":
"4baccb809a3578f4xZGp74ik1k54wD03Y25fJ7KRIR7oVJsfgyFuQj4yaCFkhQMA5qjHyETS95PCy3uKcNm2bz6NrtVFHELWPh
    "identifierInProvider": 11217524525076510964,
    "creationDate": "2020-08-18T16:02:26.0136931+00:00"
  }
}
```


API key can be used instead of the access token in the `Authorization` headers of all further requests.

```
Authorization: Bearer
4baccb809a3578f4xZGp74ik1k54wD03Y25fJ7KRIR7oVJsfgyFuQj4yaCFkhQMA5qjHyETS95PCy3uKcNm2bz6NrtVFHELWPh7mZx
```

You can temporarily restrict access with the issued API key. To do that, set the `disabled` value for the `status` property of the user identity resource associated with that API key. For details on how to modify user identities, see Modify User Identity.

If you want to disable access to REST API with the issued API key permanently, you can delete the user identity resource associated with that API key. For details, see Delete User Identity.

HTTP Methods

To perform operations with Veeam Service Provider Console entities using REST API, you can use the following standard HTTP methods:

- `GET` — retrieves information about a resource or collection.
- `POST` — creates a new resource in a collection or activates a resource action.
- `PATCH` — makes changes to a resource.
- `PUT` — replaces a resource.
- `DELETE` — removes a resource from a collection.

POST Method

The POST HTTP method can be used in two different ways:

To create a new resource, for example, a new location. To perform an operation, for example, undo a failover plan or update a license.

Post New Resource

All requests that create a new resource require a request body. The body must contain properties for this resource. For example, to create a new location, send the following request:

```
Request:
POST https://<hostname>:1280/api/v3/organizations/locations

Request Header:
Content-Type: application/json
Authorization: Bearer <Access-Token>
```

```
Request Body:
{
  "organizationUid": "89d3b0c2-529f-4d51-988a-d8ae92f3259b",
  "name": "Default location",
  "quotaGb": 10,
  "isDefault": false
}
```

Post Action

Requests that perform an action do not require a request body. For example, to undo a failover plan, send the following request:

```
Request:
POST https://<hostname>:1280/api/v3/failoverPlans/{planUid}/undo

Request Header:
Content-Type: application/json
Authorization: Bearer <Access-Token>
```

In case of success, the POST HTTP method returns the *200 OK* response code.

PATCH Method

The PATCH HTTP method is used to modify the values of the resource properties.

The PATCH HTTP method requires a request body. The body of the request must contain representation of the JSON Patch operations that you want to perform on the resource. For details on available operations and structure of a request body, see RFC 6902.

For example, the following request changes maximum number of workstations that a reseller can manage.

```
Request:
PATCH https://localhost:1280/api/v3/organizations/resellers/debcdb55-afbd-4772-bf99-5ea6a3249f43

Request Header:
Authorization: Bearer <Access-Token>
Content-Type: application/json

Request Body:
{
  "op": "replace",
  "path": "/quota/workstationsQuota",
  "value": "450"
}
```

Some operations that are performed using the POST HTTP method can also be performed using the PATCH HTTP method. For example, the following request runs a job.

```
Request:
PATCH https://<hostname>:1280/api/v3/infrastructure/backupServers/jobs/14513169-46c6-4ee1-9619-
f218ada245a9

Request Header:
Authorization: Bearer <Access-Token>
Content-Type: application/json

Request Body:
{
  "op": "replace",
  "path": "/status",
  "value": "Running"
}
```

In case of success, the PATCH HTTP method returns the *200 OK* response code.

Errors

In the Veeam Service Provider Console REST API, errors are returned with the following HTTP status codes:

- All client errors (*400-499*) are returned with the associated HTTP status codes.
- All server errors (≥*500*) are returned with the *520* HTTP status code.

The body of the error response contains detailed information about error specifics. The following example illustrates the response for the *400 Bad Request* error.

```
{
  "errors": [
    {
      "message": "Property value must be larger or equal to 0.",
      "type": "logical",
      "code": 400,
      "parameterName": "offset"
    }
  ]
}
```

where:

- `message` — short description of the problem.
- `type` — error type. For details on error types, see Errors Types.
- `code` — error code. This code may differ from the HTTP status code. For a list of error codes, see Error Response Codes.

- `parameterName` — name of the parameter that has invalid value. If an error is caused by a wrong property value, `parameterName` is replaced with `propertyName`.

Error Types

All errors in the Veeam Service Provider Console REST API are classified into the following types:

- `transport` errors occur if Veeam Service Provider Console server fails to receive requests. You can fix these errors by sending the request again.
- `logical` errors are caused by business logic issues on the client side.
- `retryablelogical` errors are caused by business logic issues on the server side and can be fixed by sending the request again.
- `security` errors are caused by authorization or authentication issues.
- `unspecified` errors occur for various reasons that are not categorized.

Error Response Codes

The following table lists all error response codes in the Veeam Service Provider Console REST API.

Code	Description
400	Invalid data input. Error response contains `propertyName` or `parameterName` property, that provides the name of a request property that caused the error.
401	To perform the requested operation, authorization is required.
403	Access to the requested resource is forbidden.

403	Access to the requested resource is forbidden.
404	Requested resource does not exist. `message` property of an error response contains the name of the missing resource.
409	Creation or modification of a resource cannot be performed due to conflicts with a resource that already exists.
415	To perform the requested operation, a different content type is required.
429	Number of requests exceeds throttling limitations. For details on throttling limitations, see Throttling Settings.
500	Unspecified cause of error.
504	REST API failed to access Veeam Service Provider Console server.
1000	Requested operation cannot be performed at the moment. `message` property of an error response contains the information about a cause of limitation.
1001	JSON Patch modifications failed to apply.
1050	Requested task has been canceled by a user.
1051	Requested task took too long to execute.
1100	Specified user does not exist.
1101	Requested operation requires functionality that is disabled by a service provider.
1200	Specified identity provider does not exist.
1201	User identity cannot be assigned to the specified user.

1202	Provided key is already assigned to a different user.
1203	Provided key is protected with password and cannot be used in the requested operation.
1204	Non-RSA keys are not supported for asymmetric authentication.
1205	Format of the provided key is not supported.
1206	Specified identity provider is disabled.
1207	Specified TOTP code is invalid.
1208	Specified MFA answer is invalid.
1209	The specified scope cannot be assigned.
1210	Container does not include a private key.
1250	IdP with the specified name already exists.
1251	Operations on IdP are unavailable while Veeam Service Provider Console portal is in the maintenance mode.
1252	IdP with the specified display name is already configured for the organization.
1253	Veeam Service Provider Console failed to receive metadata from IdP.
1254	Veeam Service Provider Console failed to process metadata received from IdP.

1255	IdP creation or modification failed.
1300	Requested operation on the specified license failed to execute.
1301	Requested operation on the specified license is already in progress.
1500	Specified location cannot be deleted because a company must have at least one assigned location.
3000	Deployment task failed to execute.
3001	Deployment task initiated by the request already exists.
4000	Version of a managed agent is outdated.
4001	Veeam CBT driver does not support the guest OS.
4002	Management agent is busy and cannot perform the requested operation.
5000	Version of a Veeam backup agent is outdated.
5001	Veeam backup agent cannot be patched or updated because it is already up-to-date.
5002	Veeam backup agent cannot be found right now and a data collection session is initiated to find that agent.
5003	Veeam backup agent cannot be found on a computer.
5004	Veeam Agent Management mode is currently changing and is not yet set to managed.
5005	Veeam backup agent is not managed by Veeam Service Provider Console.

5006	Veeam CBT driver is not compatible with the *Workstation* agent mode.
5007	Veeam CBT driver cannot be installed due to outdated Veeam backup agent version.
5008	Veeam CBT driver is already installed on the computer.
5009	Veeam CBT driver is not installed and cannot be deleted.
6000	No job or policy is configured for the Veeam backup agent.
7000	Veeam ONE version is outdated.
9000	Failover to a future date cannot be performed.

Extensions

Schema extensions are the expressions that can be found in the Veeam Service Provider Console REST API specification and that provide additional information on how extended operations function. The following table lists the current set of extensions.

Extension	Example Value	Description
x-veeam-vspc-patch-for	`"$ref": "#/definitions/UserLogin"`	Is specified for PATCH operations. Refers to a resource that can be modified by the PATCH method.
x-veeam-vspc-get-for	`"$ref": "#/definitions/`	Is specified for GET operations that require to

	`User"`	provide UID. Refers to a resource collection that contains a resource with the specified UID.
x-ms-enum	`"name": "AlarmCategoryType"`	Indicates that the enum name can be changed during code generation.
x-veeam-expandable	`"true"`	Indicates that the extended operation can return additional properties.
x-veeam-allow-anonymous	`"true"`	Indicates that the extended operation does not require authorization.
x-veeam-nullable	`"true"`	Indicates that properties in the extended schema can have null values.
x-veeam-vspc-deny-async-action-registration	`"true"`	Indicates that an operation cannot be executed asynchronously. For details, see Asynchronous Processing.
x-veeam-pagination	`"true"`	Indicates that pagination is enabled for the extended operation. For details on pagination, see Pagination.
x-veeam-pagination-limit	`"true"`	Indicates that the `limit` query parameter is enabled for the extended operation. For details, see Limit Parameter.
x-veeam-pagination-offset	`"true"`	Indicates that the `offset` query parameter is enabled for the extended operation. For details, see Offset Parameter.

x-veeam-pagination-filter	`"true"`	Indicates that the `filter` query parameter is enabled for the extended operation. For details, see Filter Parameter.
x-veeam-pagination-sort	`"true"`	Indicates that the `sort` query parameter is enabled for the extended operation. For details, see Sort Parameter.
x-veeam-pagination-select	`"true"`	Indicates that the `select` query parameter is enabled for the extended operation. For details, see Select Parameter.
x-veeam-vspc-admitted-roles	`"PortalAdministrator"`, `"CompanyOwner"`, `"ResellerOwner"`, `"ResellerAdministrator"`, `"CompanyAdministrator"`	Array of user roles to whom the extended operation is available.
x-extensible-enum	`"true"`	Indicates that new values can appear in the extended enum. If client version of the Veeam Service Provider Console REST API does not support new values, these values are displayed as `Unknown`.
x-veeam-fire-and-forget	`"true"`	Indicates that the extended operation returns a positive response if a request is received. Execution of requested operation cannot be guaranteed.

x-veeam-create-by-default	`"true"`	Indicates that if the extended property is not specified while the resource is created, it will have a default value.
x-veeam-empty-array-by-default	`"true"`	Indicates that if the extended array is not specified while the resource is created, it will be generated empty by default.

Multi-Factor Authentication

You can configure multi-factor authentication (MFA) for additional security of user accounts. In Veeam Service Provider Console, this authentication method is based on the TOTP algorithm and requires a user to install an authenticator application on the trusted device. Veeam Service Provider Console supports all applications that use TOTP mechanism, however we recommend Google Authenticator.

You can enable and disable MFA for a user account using REST API.

> **IMPORTANT**
>
> After you enable MFA for a user, that user can no longer authenticate with user name and password to the Veeam Service Provider Console REST API. Before you assign a second authentication factor to the account that requires access to REST API, configure authorization method described in the API Key-Based Authentication section.

Configuring MFA for Own Account

To enable MFA for your user account:

1. Send the HTTP GET request to the `/authentication/keys/totp-secret` path. The server will return the following data in the response

body:

- `secret` — MFA secret.
- `secretUrl` — URL that contains MFA secret and user account data.

```
Request:
POST https://<hostname>:1280/api/v3/authentication/keys/totp-secret

Request Headers:
Content-type: application/json
Authorization: Bearer <API-Key>

Response Code:
200 OK

Response Body:
{
  "data": {
    "secret": "3QEYAIQAKPC2A4ENSZEXHL4OQLWJ4XNC",
    "secretUrl": "otpauth://totp/vspc%5cAdministrator?
secret=3QEYAIQAKPC2A4ENSZEXHL4OQLWJ4XNC&issuer=My%20Company%20Portal"
  }
}
```

2. Install an authenticator application on the trusted device and enter the secret key to create an account.

   The application will generate a verification code.

   TIP

   You can use a QR code to create an account in the authenticator application:

a. On another device, open a QR code generator in a web browser.

b. In the QR code generator, insert the `secretUrl` link that you received at step 1. The QR code generator will display a QR code.

c. On your trusted device, open an authenticator application and scan the displayed QR code using the device camera. The application will automatically create an account and generate a six-digit verification code.

3. Send the HTTP POST request to the `/users/{userUid}/logins/totp` path, where `userUid` is UID of your user. Additionally you must provide the following query string parameters:

   - secretUrl — URL-encoded secret that you received at step 1.
   - code — active security code.
   - description — description for a new user identity.
   - scope — services that are available to a new user identity. In this operation, only the ui value is available.

   The server will return a response containing a resource representation of a new user identity associated with MFA.

```
Request:
POST https://<hostname>:1280/api/v3/users/955b0b80-03b2-49fd-bf38-14785ecff0a8/logins/totp?
secretUrl=otpauth://totp/vspc%5cAdministrator?
secret=3QEYAIQAKPC2A4ENSZEXHL4OQLWJ4XNC&issuer=My%20Company%20Portal&code=324568&description=MFA
for Alpha Administrator&scope=ui

Request Headers:
Content-type: application/json
Authorization: Bearer <API-Key>

Response Code:
200 OK
```

```
Response Body:
{
  "data": {
    "id": 12,
    "userUid": "955b0b80-03b2-49fd-bf38-14785ecff0a8",
    "userName": "alpha_admin",
    "companyId": "3b139961-6b11-4fd8-9e87-c2be1b0e9ca8",
    "companyName": "Alpha",
    "identityProviderName": "TOTP",
    "description": "MFA for Alpha Administrator",
    "identifierInProvider": "11217524525076510964",
    "scopes": [],
    "status": "enabled",
    "parameters": "",
    "creationDate": "2020-08-18T16:02:26.0136931+00:00"
  }
}
```

Enabling MFA for Other Users

To enable MFA for other users, send the HTTP PATCH the following request to the `/users/{userUid}` path, where `{userUid}` is UID of a user:

```
[
  {
    "op": "replace",
    "path": "mfaPolicyStatus",
    "value": "enabled",
  }
]
```

Enabled MFA only affects users that access Veeam Service Provider Console Portal. On the next authorization, these users will be prompted to configure MFA. Authentication to Veeam Service Provider Console REST API remains intact.

Disabling MFA

To disable MFA for other users, send the HTTP PATCH the following request to the `/users/{userUid}` path, where `{userUid}` is UID of a user:

```
[
  {
    "op": "replace",
    "path": "mfaPolicyStatus",
    "value": "disabled",
  }
]
```

If you want to permanently disable MFA for a specific user, you can delete the associated user identity resource. For details, see Delete User Identity.

Asynchronous Processing

In some cases requests take significant time to process or there are multiple requests processed at the same time.The Veeam Service Provider Console REST API v3 may handle such requests asynchronously. Asynchronous processing decreases the load on web-browsers and notifies clients when their requests are accepted but not yet processed by the server.

When a request initiates an async action, a user receives a response with the *202 Accepted* code. The Location header of the response contains a link to the `/asyncAction/{asyncActionId}` resource of the task. This link allows the user who initiated the task to track its progress. For details, see Get Async Action.

If you want to assign a specific UID to an async action, add the `x-request-id` header with this UID as a value to the request.

NOTE

- Operations marked with the `x-veeam-vspc-deny-async-action-registration` extension cannot be handled asynchronously.
- Some operations can take longer time to finish but do not behave as async actions. These operations are marked with the `x-veeam-vspc-fire-and-forget: true` extension.

Async Action Settings

Results of an async action are stored on a server for a limited period of time. You can change duration of this time period either on the REST API site and on the server site.

To configure async action timeout on the REST API site, on the Veeam Service Provider Console server, go to the `\Veeam\Availability Console\Web UI` directory and open the `appsettings.json` file. The `AsyncAction` section of the file contains the following parameters:

- `ResultExpirationTime` — time period during which results of successfully processed actions are kept. The default value is `5` minutes.
- `FailureExpirationTime` — time period during which error results of actions are kept. The default value is `1` hour.
- `TimerResolution` — interval between action status checks. Defines the smallest unit for other async action parameters. The default value is `1` minute.

To apply changes, do one of the following:

- In Task Manager, end the `Veeam.AC.WebUI` task.
- In Internet Information Services, recycle the `Veeam Service Provider Console Web UI` application pool.

NOTE

While configuring your application to perform action status checks, set check interval that exceeds half of the result retention time.

Request Proxying

Request proxying allows you to redirect HTTP requests to Veeam products using Veeam Service Provider Console. It can be beneficial in case Veeam Service Provider Console does not support required features available in a target Veeam product.

Currently you can proxy requests to Veeam Backup & Replication, Veeam ONE and Veeam Backup for Microsoft 365.

You can start proxying requests after Veeam Service Provider Console agent is deployed on a machine where a Veeam product is installed and product data is collected.

Authorization

REST API proxying does not require additional authorization.

Note that proxying provides a user with maximum privileges in a target Veeam product. To prevent read-only users from applying any changes to target product, users authorized with read-only API keys are permitted to send only the GET requests.

URL Paths

To proxy a request, provide URL path targeting required Veeam product in the following format: `/proxy/{managementAgentUid}/{product}/{targetProductEndpointPath}`.

For example, the following URL path can be used to receive Veeam Backup for Microsoft 365 server jobs:

```
https://localhost:1280/proxy/69ca6904-25fd-40e8-86a6-aa16a8bf7723/vb365/v7/jobs
```

Proxy Sessions

When you proxy a request, Veeam Service Provider Console Service sends the `PrepareProxySession` request to the Veeam Service Provider Console management agent installed on a Veeam product server. The management agent acquires Veeam product base URL and API token and starts a proxy session which is a set up connection of Veeam Service Provider Console Service and Veeam product API.

By default, if a proxy session is not being used for 15 minutes, API token expires and Veeam Service Provider Console Service drops the session. In this case the next time a user sends a proxied request, the management agent first sends a request to acquire a new token and start a new session, and then sends the user request to a Veeam product API.

This approach helps proxying process by maintaining system stability in situations when Veeam Service Provider Console Service or management agent becomes unavailable for some time.

Responses

Normally Veeam Service Provider Console applies no changes to proxied request responses. However, if a Veeam product access token expires and a management agent receives an error response with the *401* HTTP status code, this management agent automatically refreshes the token and sends the request again. Note that if a user receives an error response with the *401* HTTP status code to the proxied request, it means that an access token is expired on the Veeam Servce Provider Console side and the user must refresh it.

If during the proxying process an issue occurs with a request or Veeam Service Provider Console infrastructure components, a user receives an error response with the *502* HTTP status code.

Obtaining Available Veeam Products

You can receive collection of Veeam products available for request proxying that are installed on a managed server. To do that, send GET HTTP request to the following path:

```
https://<hostname>:1280/api/v3/infrastructure/managementAgents/{managementAgentUid}/
```

```
proxyableProducts
```

where `managementAgentUid` is a UID assigned to a management agent installed on the server where Veeam products reside.

The response contains an array of Veeam product resources. Each resource has the following properties:

- `ProxyProduct` - Veeam product that accepts proxied requests.
- `ProxyProductUrlRepresentation` - part of URL path that must be used to proxy requests to a Veeam product.

Logging

Veeam Service Provider Console logs proxying errors in all the infrastructure components (WebUI, Service, Agent). The `ProxySessions.log` file resides on the server where the `Service` component is installed. The default minimum log level is `Information` which includes all basic log entries, for example, proxying errors, basic proxying messages about opened sessions, resuming paused sessions, closed sessions and so on.

You can include additional information like used method, user UID of a caller, URL and so on by enabling verbose logging. To do that, open the `appSettings.Logging.json` file and replace the `Information` value of the `MinimumLevel` parameter with `Verbose`.

Configuration Keys

You can provide the following configuration keys in the `configuration.overrides.json` file to modify default proxying settings.

- `HttpProxy_SessionsCleanupInterval` - time interval between idle sessions cleanups. The default value is `1` minute.
- `HttpProxy_SessionIdleTimeout` - maximum time a proxy session can stay idle before it gets dropped. The default value is `15` minutes.
- `HttpProxy_SessionPausedTimeout` - time interval after which an idle proxy session is considered paused. A paused session is resumed if a user sends a new request. The default value is `5` minutes.

Versioning

Veeam Service Provider Console REST API is available in multiple versions. To work with resources available in a particular REST API version, you must specify it in the `x-client-version` header of each request.

The following table lists REST API versions and their current status.

REST API Version	Support Status	Veeam Service Provider Console Version
3.6	Supported	9.0
3.5.1	Supported	8.1
3.5	Supported	8.1
3.4	Supported	8.0
3.3	Not supported	7.0
3.2	Not supported	6.0
3.1	Not supported	5.0
3.0	Not supported	4.0
2	Not supported	2.0, 2.0 Update 1, 3.0

Some enums are marked with the `x-extensible-enum: true` extension. For those enums, new possible values may be added in further product

Some enums are marked with the `x-extensible-enum: true` extension. For those enums, new possible values may be added in further product updates.

To avoid future issues with your integration, you can send the `x-client-version` header in each request to return the `unknown` value instead of the values, that your client does not support. The value of the header is the version of the Veeam Service Provider Console REST API for which the integration is configured.

Throttling

To avoid excessive loading of Veeam Service Provider Console server caused by processing too many requests simultaneously or during a specific time period, the Veeam Service Provider Console REST API controls request rate through throttling mechanism.

Each request is classified as a part of one or more of the following groups:

- All requests
- Requests with specific `UserId` parameter value
- Requests with specific `IpAddress` parameter value
- Requests with specific `UserID` and `IpAddress` parameter values

According to throttling policy of those groups, a request is processed if it fits both of the following limitations:

- Maximum number of requests processed simultaneously
- Maximum number of requests processed during a unit of time

If at least one of those limitations is exceeded, server returns error with the *429* code. If maximum number of requests processed during a specific time period is exceeded, the response additionally may contain the `retryAfter` property that has the value equal to the number of seconds till the next time period begins:

```
{
```

```
  "errors": [
    {
      "type": "transport",
      "code": 429,
      "retryAfter": 25
     }
  ]
}
```

Throttling Settings

If your infrastructure can handle more requests than it is allowed by default throttling policy, you can increase values of the limitations. To do that, in the machine on which the Veeam Service Provider Console Web UI component is installed, go to the `\Veeam\Availability Console\Web UI` directory and open the `appsettings.json` file. The `Throttling` section of the file contains the following parameters:

- `IsDisabled` — defines whether throttling is disabled. By default has the `false` value.
- `MaxParallelRequestsMultiplier` — multiplier for the maximum number of parallel requests. The default value is `1`.
- `MaxRequestPerTimeUnitMultiplier` — multiplier for the maximum number of requests processed during a unit of time. By default is equal to `1`.
- `SkipIpAddresses` — list of comma-separated IP addresses to whose requests custom settings are not applied.

Additionally you can provide the `SkipIpNetworks` parameter with array of CIDR notation masks of networks to whose requests custom settings are not applied.

The following example shows a modified configuration:

```
"Throttling": {
  "IsDisabled": false,
```

```
  "MaxParallelRequestsMultiplier": "5",
  "MaxRequestPerTimeUnitMultiplier": "5",
  "SkipIpAddresses": "192.168.0.1,192.168.0.2",  "SkipIpNetworks":
["192.10.100.14/24","198.126.10.0/22"]
},
```

To apply changes, do one of the following:

- In Task Manager, end the `Veeam.AC.WebUI` task.
- In Internet Information Services, recycle the `Veeam Service Provider Console Web UI` application pool.

Query Parameters

The Veeam Service Provider Console REST API supports query string parameters that you can specify to control the amount and order of data that the Veeam Service Provider Console REST API returns for a resource or collection.

The Veeam Service Provider Console REST API supports the following query parameters:

- Offset Parameter
- Limit Parameter
- Sort Parameters
- Filter Parameter
- Select Parameter
- Expand Parameter

Offset Parameter

The `offset` query parameter is used to exclude from a response the first N items of a resource collection.

For example, to show in a response all users starting from the forth one, send the following request:

```
GET https://<hostname>:1280/api/v3/users?offset=3
```

You can combine the limit and the `offset` options to request a particular set of items. Note that the `offset` option is applied before the `limit` option, regardless of its position in the request. That is, top results are selected from a collection where a set of items is already excluded.

For example, to return the second and the third backup repositories available in Veeam Service Provider Console, send the following request:

```
GET https://<hostname>:1280/api/v3/infrastructure/backupServers/repositories?offset=1&limit=2
```

The Veeam Service Provider Console REST API returns resources in the default order based on its own semantics.

Limit Parameter

The `limit` query parameter specifies the number of resources that a single response page contains.

For example, to show in a response only the first three backup servers from the `/backupServers` collection, send the following request:

```
GET https://<hostname>:1280/api/v3/infrastructure/backupServers?limit=3
```

Note that the `limit` parameter value cannot surpass the maximum number of elements on page which is *500* by default. For details, see Pagination.

You can combine the `limit` and the offset parameter to request a particular set of items. Note that the `offset` parameter is applied before the `limit` parameter, regardless of its position in the request. That is, top results are selected from a collection where a set of items is already excluded.

For example, to return 5 companies that follow the first 10 companies available in Veeam Service Provider Console, send the following request:

```
GET https://<hostname>:1280/api/v3/organizations/companies?offset=10&limit=5
```

The Veeam Service Provider Console REST API returns resources in the default order based on its own semantics.

Sort Parameters

The `sort` query parameter is used to sort items in a resource collection returned in a response.

The parameter accepts the *array* data type. To sort a resource collection, use the `sort` query parameter in the following format:

```
GET https://<hostname>:1280/api/v3/<endpoint>?
sort=[{"property":"<property>","direction":"<direction>","collation":"<collation>"}]
```

where:

- `endpoint` is a resource collection endpoint.
- `property` is the searched property name.

  NOTE

  To sort a resource collection based on a property of a resource subschema, use the `<subschema>.<property>` format to specify the resource property name where `<subschema>` is the subschema name.

- `direction` is a direction specifier. The following values are available:
  - *ascending*

    - *ascending*
    - *descending*

- `collation` is the type of rules that determine how sorted values are compared to each other. The following types are available:
    - *ordinal* — compare strings symbol-by-symbol
    - *ignorecase* — compare strings symbol-by-symbol ignoring character case
    - *lexicographic* — compare words with words and numbers with numbers

  The default value is *ordinal*.

Examples

To sort company cloud backup resources by used storage quota, send the following request:

```
GET https://vspc:1280/api/v3/organizations/companies/sites/backupResources/usage?
sort=[{"property":"usedStorageQuota","direction":"descending","collation":"lexicographic"}]
```

To sort subscription plans by name type and currency, send the following request:

```
GET https://vspc:1280/api/v3/subscriptionPlans?
sort=[{"property":"type","direction":"ascending"},
{"property":"currency","direction":"ascending"}]
```

Filter Parameter

The `filter` query parameter is used to filter items in a resource collection to return a subset of resources in a response. The subset includes only those resources that satisfy parameter value specified in the query.

To filter a resource collection use the `filter` query parameter in the following format:

```
GET https://<hostname>:1280/api/v3/<endpoint>?
filter=[{"property":"<property>","operation":"<operation>","collation":"<collation>","value":"<value>"
```

where:

- `endpoint` is a resource collection endpoint.
- `property` is the searched property name.

  NOTE

  To filter a resource collection based on a property of a resource subschema, use the `<subschema>.<property>` format to specify the resource property name where `<subschema>` is the subschema name.

  - `operation` is a relational or logical operator.
  - `collation` is the type of rules that determine how specified values are compared with resource property values. The following types are available:
    - *ordinal* — compare strings symbol-by-symbol
    - *ignorecase* — compare strings symbol-by-symbol ignoring character case
    - *lexicographic* — compare words with words and numbers with numbers

  The default value is *ordinal*.

  - `value` is a resource property value.

The Veeam Service Provider Console REST API supports the following operators:

Operator	Description
in	Returns resources that match at least one of the values specified for a property.
contains	Returns resources that contain the value specified for a property.
subset	Returns resources that contain a subset of values specified for a property.
superset	Returns resources that contain a superset of values specified for a property.
equals	Returns resources that match the specified property value.
notEquals	Returns resources that do not match the specified property value.
lessThan	Returns resources with the property value that is less than the specified value.
lessThanOrEqual	Returns resources with the property value that is less than or equal to the specified value.
greaterThan	Returns resources with the property value that is greater than the specified value.
greaterThanOrEqual	Returns resources with the property value that is greater than or equal to the specified value.

To group multiple filter expressions, you can include them into a grouping expression. A grouping expression includes the following elements:

- `operation` is a relational or logical operator.

- `items` is an array of filter expressions.

A grouping expression has the following format:

```
GET https://<hostname>:1280/api/v3/<endpoint>?filter=[{"operation":"<operation>","items":
[{"property":"<property>","operation":"<operation>","value":"<value>","collation":"<collation>"},
{"property":"<property>","operation":"<operation>","value":"<value>","collation":"<collation>"}]}]
```

To combine filter expressions you can use the following operators:

Operator	Description
exclusiveOr	Returns resources that match one of the filter expressions.
or	Returns resources that match one or more of the filter expressions.
and	Returns resources that match all filter expressions.
not	Returns resources that do not match all filter expressions.

Examples

To get all triggered alarms with the *Error* status, send the following request:

```
GET https://vspc:1280/api/v3/alarms/active?
filter=[{"property":"lastActivation.status","operation":"equals","collation":"ignorecase","value":"erro
```

To get all users with the Portal Operator and Portal Administrator roles:

```
GET https://vspc:1280/api/v3/users?filter=[{"operation":"or","items":
[{"property":"role","operation":"equals","value":"PortalAdministrator"},
{"property":"role","operation":"equals","value":"PortalOperator"}]}]
```

Select Parameter

The `select` parameter is used to return only selected resource properties in a response.

The parameter accepts the *array* data type. To return selected properties, use the `select` parameter in the following format:

```
GET https://<hostname>:1280/api/v3/<endpoint>?select=[{"propertyPath":"<property>"}]
```

where:

- `endpoint` is a resource collection endpoint.
- `property` is the searched property name.

> **NOTE**
>
> To filter a resource collection based on a property of a resource subschema, use the `<subschema>.<property>` format to specify the resource property name where `<subschema>` is the subschema name.

For example, to return the `/organizations/resellers` collection with only reseller UID, name and status displayed, send the following request:

```
GET https://vspc:1280/api/v3/organizations/resellers?select=[{"propertyPath":"instanceUid"},
```

{"propertyPath":"name"},{"propertyPath":"status"}]

Expand Parameter

The `expand` query parameter allows you to add embedded property data to a response.

Embedded property is a property that contains resource representations of the related entities. In Veeam Service Provider Console REST API, these are root entities which usually represent the same object but from a different perspective. For example, organization is the root entity of a company. By including embedded properties in the `GET /organizations/companies` request, you can receive information both on all companies and organizations related to those companies without sending additional requests.

The following example response demonstrates a collection of company resources with embedded properties included.


```
{
  "meta": {
    "pagingInfo": {
      "total": 2,
      "count": 2,
      "offset": 0
    }
  },
  "data": [
    {
      "instanceUid": "eb866e8f-a0f5-41f4-9dd9-8cc59266ff25",
      "name": "Alpha",
      "status": "Active",
      "resellerUid": "a1797b02-364e-4e31-b052-7d7db48edb4a",
      "subscriptionPlanUid": "615808c7-12ba-4cd4-a1c2-3ba78d95fbe9",
      "isRestAccessEnabled": true,
      "isAlarmDetectEnabled": true,
      "companyServices": {
```

```
    "hostedServices": {
      "isVbPublicCloudManagementEnabled": false
    },
    "remoteServices": {
      "isBackupResourcesEnabled": true,
      "backupAgentsManagement": {
        "workstationAgentsQuota": 20,
        "serverAgentsQuota": 20
      },
      "backupServersManagement": {
        "backupServerQuota": 20
      },
      "vb365ServersManagement": null,
      "isVbPublicCloudManagementEnabled": false
    }
  },
  "loginUrl": null,
  "ownerCredentials": {
    "userName": "alphauser",
    "password": null
  },
  "_embedded": {
    "organization": {
      "instanceUid": "ece6d725-2fc2-48e6-af85-b78b3778b63c",
      "name": "Alpha",
      "alias": "alpha",
      "type": "Company",
      "taxId": "65432",
      "email": "t.barb@alpha.com",
      "phone": "916-336-1534",
      "country": 1,
      "state": 5,
      "countryName": "USA",
      "regionName": "West",
```

```
      "city": "Sacramento",
      "street": "Awesome street",
      "locationAdmin0Code": "us",
      "locationAdmin1Code": "us-ca",
      "locationAdmin2Code": null,
      "notes": "",
      "zipCode": "95814",
      "website": "www.alpha.com",
      "veeamTenantId": "10"
    }
  }
},
{
  "instanceUid": "279595ca-2027-4110-abc8-63619d14e25a",
  "name": "Beta",
  "status": "Active",
  "resellerUid": "a1797b02-364e-4e31-b052-7d7db48edb4a",
  "subscriptionPlanUid": null,
  "isRestAccessEnabled": true,
  "isAlarmDetectEnabled": false,
  "companyServices": {
    "hostedServices": {
      "isVbPublicCloudManagementEnabled": false
    },
    "remoteServices": {
      "isBackupResourcesEnabled": true,
      "backupAgentsManagement": {
        "workstationAgentsQuota": 20,
        "serverAgentsQuota": 20
      },
      "backupServersManagement": {
        "backupServerQuota": 20
      },
      "vb365ServersManagement": null,
```

```
      "isVbPublicCloudManagementEnabled": false
    }
  },
  "loginUrl": null,
  "ownerCredentials": {
    "userName": "betauser",
    "password": null
  },
  "_embedded": {
    "organization": {
      "instanceUid": "61a6c733-d6a9-4540-9698-1d4e3c6bdaab",
      "name": "Beta",
      "alias": "beta",
      "type": "Company",
      "taxId": "09876",
      "email": "m.lore@beta.com",
      "phone": "503-254-0212",
      "country": 1,
      "state": 37,
      "countryName": "USA",
      "regionName": "West",
      "city": "Portland",
      "street": "5006 Gateway Road",
      "locationAdmin0Code": "us",
      "locationAdmin1Code": "us-or",
      "locationAdmin2Code": null,
      "notes": "",
      "zipCode": "97220",
      "website": "www.beta.com",
      "veeamTenantId": "2"
    }
  }
 }
]
}
```

To include embedded property content, provide the name of a root entity as a value of the `expand` parameter. The following example request returns the root discovery rules of the discovery rules for Microsoft Windows computers.

```
GET https://<hostname>:1280/api/v3/discovery/rules/windows?expand=DiscoveryRule
```

NOTE

PATCH HTTP method cannot be performed on embedded properties.

Pagination

To increase readability and optimize network traffic, the Veeam Service Provider Console REST API applies pagination to a number of responses that contain array of values. Operations that return responses with pagination are marked with the `x-veeam-pagination: true` extension in the specification.

Pagination data is returned in the `pagingInfo` subsection of the `meta` section of a response and includes the following properties:

- `total` — total number of resources returned.
- `count` — number of resources on the current page. The default value is `100`. You can modify this number using the limit parameter.
- `offset` — number of resources skipped. You can skip resources using the offset parameter.

NOTE

If the number of resources in a collection changes, some resources can move from one page to another.

By default, the maximum number of elements on a page that you can set using the `limit` parameter is limited to `500`. To increase that number:

1. On the Veeam Service Provider Console server, go to the `\Veeam\Availablity Console\Web UI` directory and open the `appsettings.json` file.
2. In the `MaxPageSize` parameter value, specify the desired maximum number of resources on a page.

   Note that the maximum value is `2147483647`.
3. To apply changes, do one of the following:
   - In Task Manager, end the `Veeam.AC.WebUI` task.
   - In Internet Information Services, recycle the `Veeam Service Provider Console Web UI` application pool.

Integration with Grafana

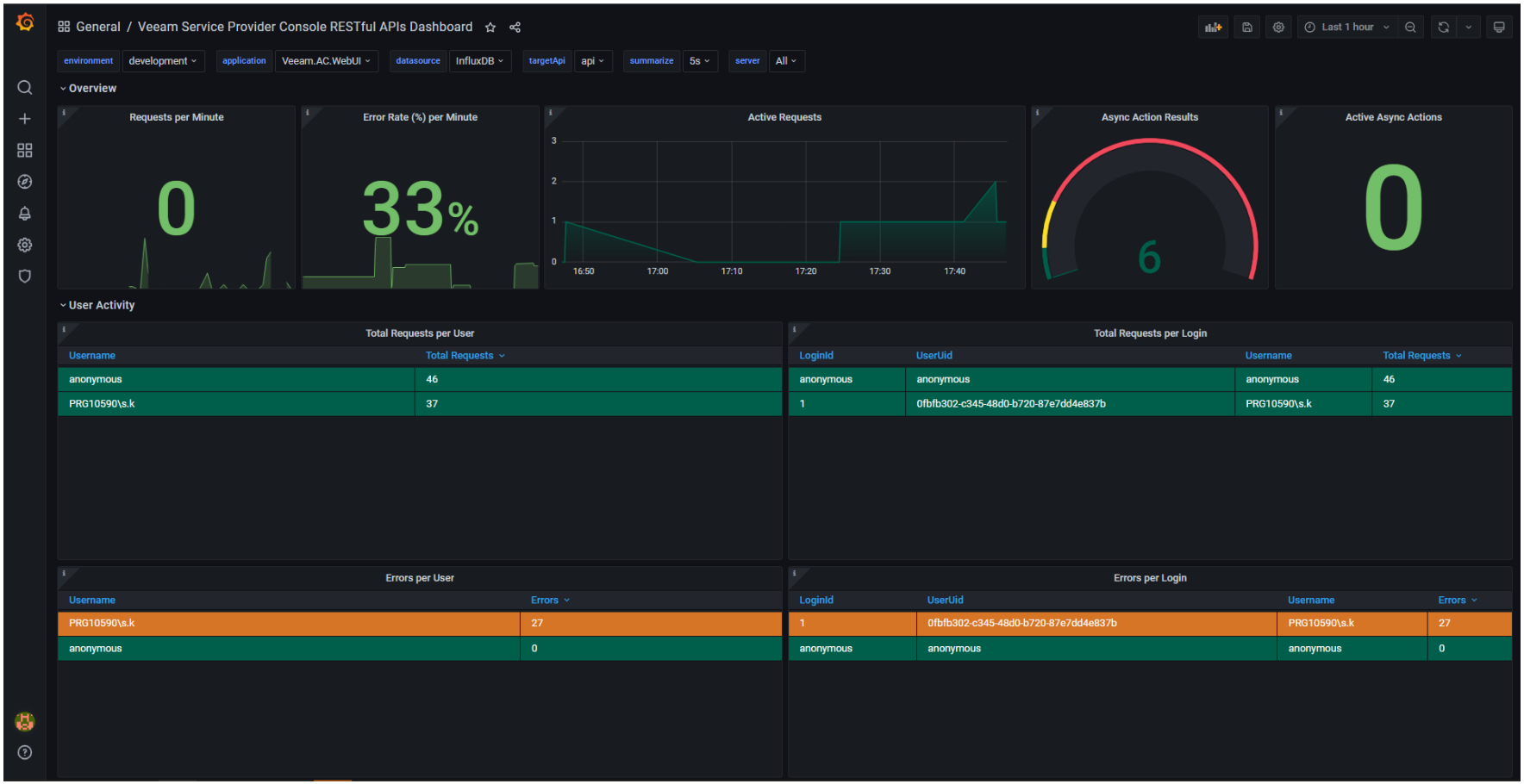
Grafana is a monitoring solution that fetches metrics and represents them in the form of dashboards. Integration with Grafana allows you to monitor the real time Veeam Service Provider Console statistics.

Veeam Service Provider Console server collects the following types of metrics that the Grafana integration utilizes:

- REST API metrics contain statistics of REST API requests sent to a Veeam Service Provider Console server. REST API metrics are represented in the *Veeam Service Provider Console RESTful APIs Dashboard* predefined dashboard.
- Veeam Service Provider Console server metrics contain server performance and infrastructure statistics data collected from the Veeam Service Provider Console server and management agents. This data is stored inside InfluxDB logical containers called measurements in the form of the Prometheus metrics. The metrics can be grouped by tags that consist of key-value pairs. Veeam Service Provider Console server metrics are represented in the following predefined dashboards:

- *Veeam Service Provider Console Backend Performance Dashboard* — contains server performance data.
- *Veeam Service Provider Console Backend Statistic Dashboard* — contains data on the managed infrastructure statistics.
- *Veeam Service Provider Console Plugins Dashboard* — contains data on the custom plugins available in Veeam Service Provider Console.

For details on the predefined dashboards and included metrics, see Preconfigured Dashboards.



Configuring Integration

Before you set up the integration, make sure that the machine on which the Veeam Service Provider Console Web UI component is installed meets the

Grafana requirements.

To collect data, Grafana requires Veeam Service Provider Console to store information about received API requests in the InfluxDB database. Note that only InfluxDB v2 or later is supported.

To view the predefined dashboards in Grafana, you must first download JSON files that contain dashboard configuration. To do that:

1. Log in to Veeam Service Provider Console.

   For details, see Accessing Veeam Service Provider Console.

2. At the top right corner of the Veeam Service Provider Console window, click **Configuration**.

3. In the configuration menu on the left, click **Plugin Library**.

4. Click the **Grafana Labs** plugin tile. The integration instruction window will open.

5. Under the step 5 of the instruction, click the links to download dashboard configuration files.

Configuring InfluxDB Integration

To configure integration with Grafana using InfluxDB database:

1. Download InfluxDB on the official Download page.

2. Install InfluxDB as described in the Install InfluxDB section of the InfluxDB documentation.

3. Configure InfluxDB as described in the Get started with InfluxDB section of the InfluxDB documentation.

4. Create an InfluxDB organization as described in the Create an organization section of the InfluxDB documentation.

5. Create an organization bucket as described in the Create a bucket of the InfluxDB documentation.

6. Create an API token as described in the Create a token section of the InfluxDB documentation.

Configuring Dashboard Containing REST API Metrics

To view *Veeam Service Provider Console RESTful APIs Dashboard* that visually represents real time statistics of REST API requests sent to a Veeam Service Provider Console server:

1. On the machine on which the Veeam Service Provider Console Web UI component is installed, open the `appsettings.json` file located in the `RestAPI` folder of the Veeam Service Provider Console installation directory.
2. In the file, change property values in the `MetricsOptions` section so that the result looks like the following:

```
  "MetricsOptions": {
  "DefaultContextLabel": "VAC_REST",
  "Enabled": true,
  "ReportingEnabled": true,
  "Reporter": "InfluxDb2"
  "InfluxDb2Reporting": {
    "Url": "http://<IP address of the server on which InfluxDB is installed>:<TCP port for
InfluxDB client-server communication>",
    "Enabled": true,
    "Organization": "<name of an organization that you configured at step 4>",
    "Bucket": "<name of a bucket that you configured at step 5>",
    "Token": "<API token that you created at step 6>"
  }
},
```

3. Save the changes.
4. Do one of the following:
   - In Task Manager, end the `Veeam.AC.WebUI` task.
   - In Internet Information Services, recycle the `Veeam Service Provider Console Web UI` application pool.
5. Download and install Grafana as described in the Install Grafana on Windows section of Grafana documentation.

6. Access Grafana and add a data source as described in the Getting Started with Grafana and InfluxDB section of Grafana documentation.
7. On the side menu, select **Create** > **Import**.
8. In the **Import** window, click the **Upload .json file** button and select the `grafana-v11-web-dashboard.json` file.
9. Click the **Import** button. The dashboard will automatically open.

Configuring Dashboard Containing Veeam Service Provider Console Server Metrics

To view dashboards that visually represent Veeam Service Provider Console server metrics:

1. On the machine on which the Veeam Service Provider Console Server component is installed, open the `configuration.overrides.json` file located in the `\Veeam\Veeam Availability Console\Configuration\Service` folder.
2. In the file, change property values in the `GeneratedConfiguration` section so that the result looks like the following:

```
  "GeneratedConfiguration": {
  "Service_InfluxDb2_Url": "http://localhost:8086",
  "Service_InfluxDb2_Enabled": true,
  "Service_InfluxDb2_Organization": "veeam",
  "Service_InfluxDb2_Bucket": "vspc",
  "Service_InfluxDb2_Token": "admin"
}
```

   where:

   - `Service_InfluxDb2_Organization` is the name of the organization that you created at step 4 of the Configuring InfluxDB Integration section.
   - `Service_InfluxDb2_Bucket` is the name of the organization bucket that you created at step 5 of the Configuring InfluxDB Integration section.

- `Service_InfluxDb2_Token` is the API key that you created at step 6 of the Configuring InfluxDB Integration section.

3. Save the changes.
4. Restart the `VeeamManagementPortalSvc` service.
5. Download and install Grafana as described in the Install Grafana on Windows section of Grafana documentation.
6. Access Grafana and add a data source as described in the Getting Started with Grafana and InfluxDB section of Grafana documentation.
7. On the side menu, select **Create** > **Import**.
8. In the **Import** window, click the **Upload .json file** button and select one of the `JSON` files that you downloaded from the Grafana plugin page in Veeam Service Provider Console.
9. Click the **Import** button. The dashboard will automatically open.
10. Repeat steps 7-9 for each downloaded JSON file.

Querying Veeam Service Provider Console Server Metrics

To display data, Grafana queries Veeam Service Provider Console server metrics by fetching them from the measurements using the InfluxQL or Flux statements and optionally filtering or grouping them by tags.

You can fetch a specific metric from an external application. To do that, use a query similar to the following:

```
-- most recent value
SELECT last("vspc.statistic.alarms_total_active_count")
FROM "prometheus";
```

If you want to customize a Grafana panel with a specific metric, use a query similar to the following:

```
SELECT last("vspc.statistic.alarms_total_active_count")
FROM "prometheus"
WHERE $timeFilter
GROUP BY time($__interval)
FILL(previous);
```

Preconfigured Dashboards

This section contains all panels included in the preconfigured Grafana dashboards for Veeam Service Provider Console.

Veeam Service Provider Console RESTful APIs Dashboard

Veeam Service Provider Console RESTful APIs Dashboard consists of the following panels:

Panel	Collected Metric	Description
Requests per Minute	`vspc.rest__total_requests_with_tags`	Number of requests processed in 1 minute.
Error Rate (%) per Minute	`vspc.rest__one_minute_error_percentage_rate`	Error percentage among responses. Updates after every processed request.
Active Requests	`vspc.rest__active_requests`	Graph that represents changes in the number of requests being processed at the moment.
Async Action Results	`vspc.rest__current_uncleaned_async_action_count`	Number of async actions whose results are not yet deleted.

Active Async Actions	`vspc.rest__current_async_action_count`	Number of currently processed async actions.
Total Requests per User	`vspc.rest__total_requests_with_tags`	Number of processed requests for each user.
Total Requests per Login	`vspc.rest__total_requests_with_tags`	Number of processed requests for each user identity.
Errors per User	`vspc.rest__total_errors_with_tags`	Number of returned errors for each user.
Errors per Login	`vspc.rest__total_errors_with_tags`	Number of returned errors for each user identity.
Throttled Requests per User	`vspc.rest__total_throttled_requests_with_tags`	Number of throttled requests for each user.
Throttled Requests per Login	`vspc.rest__total_throttled_requests_with_tags`	Number of throttled requests for each user identity.
Total Requests	`vspc.rest__total_requests`	Graph that represents changes in the total number of processed requests.
Requests per Endpoint	`vspc.rest__total_requests_with_tags`	Graph that represents changes in the number of processed requests for each endpoint.
Response Time	`vspc.rest__transactions_timer`	Percentiles of response time at any second.

	`met`	
Response Time per Endpoint (95th Percentile)	`vspc.rest__transactions_per_endpoint_timer`	Graph that represents 95th percentiles of response time at any second for each endpoint.
Requests per Endpoint (Table)	`vspc.rest__total_requests_with_tags`	Table that represents the number of processed requests for each endpoint.
Response Time per Endpoint (95th Percentile Table)	`vspc.rest__transactions_per_endpoint_timer`	Table that represents 95th percentiles of response time for each endpoint.
Post Request Size	`vspc.rest__post_size`	Graph that represents changes in the percentiles of POST request sizes.
Put Request Size	`vspc.rest__put_size`	Graph that represents changes in the percentiles of PUT request sizes.
Total Errors by Code	`vspc.rest__total_errors_with_tags`	Ratio of errors for each error code.
Total Errors by Type	`vspc.rest__total_errors_with_tags`	Ratio of errors for each error type.
Total Errors	`vspc.rest__total_errors`	Graph that represents changes in the total number of errors.
Error Rate (%)	`vspc.rest__one_minute_error_percentage_rate`	Graph that represents changes in the error percentage among responses. Updates after every processed request.
Errors per User and Route	`vspc.rest__total_errors_wi`	Number of errors for each user and endpoint.

	`th_tags`	
Errors per Endpoint	`vspc.rest__total_errors_with_tags`	Number of errors with error types for each endpoint.

The following metrics are also collected and can be used to create custom panels:

- `vspc.rest__total_throttled_requests` — total number of throttled requests.
- `vspc.rest__one_minute_error_percentage_rate_per_endpoint` — error percentage among responses returned in one minute for each endpoint. Updates after every processed request.

Veeam Service Provider Console Backend Performance Dashboard

Veeam Service Provider Console Backend Performance Dashboard contains the following metrics.

Application Metrics

Application metrics include information on data collection and processing in Veeam Service Provider Console architecture. Collected Veeam Service Provider Console server data is processed in the following way:

All collected data is saved to the staging table for temporary storage. Staging table data is distributed in the database among two data types: Operational data — infrastructure current state data collected by management agents. Analytical data — infrastructure historical data collected by management agents. Information on execution time of each process is included into a related metric and represented in the following panels:

Panel	Collected Metric	Description
Incoming Analytical Data Processing Time	`vspc.performance.server.database_bp_processing_time`	Time taken to process collected analytical data.

Incoming Operational Data Processing Time	`vspc.performance.server.database_od_processing_time`	Time taken to process collected operational data.
Staging Table Populating Time	`vspc.performance.server.entity_traffic_saving_time`	Time taken to save collected data into the staging table.

These metrics may indicate presence of performance issues in an infrastructure:

- Values below `200` seconds — no apparent performance issues.
- Values over `200` seconds — signs of minor performance issues.
- Values over `600` seconds — signs of major performance issues.

The rest of application metrics do not have recommended thresholds and are represented in the following panels:

Panel	Collected Metric	Description
Incoming Analytical Data Rows Awaiting For Processing	`vspc.performance.server.database_bp_staging_rows`	Number of analytical data changes awaiting processing.
Incoming Operational Data Rows Awaiting For Processing	`vspc.performance.server.database_od_staging_rows`	Number of operational data changes awaiting processing.
Pending For Insertion Into The Staging Table (Rows)	`vspc.performance.server.entity_traffic_rows`	Rows pending for insertion into the staging table.
Pending For Insertion Into The Staging Table (Bytes)	`vspc.performance.server.entity_traffic`	Volume of data pending for insertion into the staging table in bytes.

Agent Data Traffic in Rows by Managed Agent Uid	`vspc.performance.agent.entity_traffic_rows`	Number of table rows in management agent traffic. Grouped by the `AgentId` tag.
Agent Data Traffic by Managed Agent Uid	`vspc.performance.agent.entity_traffic`	Data volume of management agent traffic. Grouped by the `AgentId` tag.

Runtime Metrics

Runtime metrics are the standard .NET runtime counters that you can use to troubleshoot performance issues.

Panel	Collected Metric	Description
Count of bytes currently in use by objects in the GC heap that haven't been collected yet	`process.runtime.dotnet.gc.objects.size`	Count of bytes currently in use by objects in the GC heap.
Committed virtual memory	`process.runtime.dotnet.gc.committed_memory.size`	Amount of committed virtual memory for the managed GC heap.
GC Count for Last Hour	`process.runtime.dotnet.gc.collections.count`	Number of garbage collections in the last hour (derivative on the panel).
Allocated on the managed GC heap for the Last Hour	`process.runtime.dotnet.gc.allocations.size`	Number of bytes allocated on the managed GC heap for the last hour (derivative on the panel).
Heap Size	`process.runtime.dotnet.gc.heap.size`	Heap size including fragmentation, as observed during the latest garbage collection.

The heap fragmentation	`process.runtime.dotnet.gc.heap.fragmentation.size`	Heap fragmentation, as observed during last GC.
GC Pause for the Last Hour	`process.runtime.dotnet.gc.duration`	Time paused in GC for the last hour (derivative on the panel).
Exceptions count for the Last Hour	`process.runtime.dotnet.exceptions.count`	Count of exceptions thrown in managed code for the last hour (derivative on the panel).
Active Timers Count	`process.runtime.dotnet.timer.count`	Number of timer instances currently active.
Thread Pool Queue	`process.runtime.dotnet.thread_pool.queue.length`	Number of work items currently queued to the thread pool.
Thread Pool Threads Count	`process.runtime.dotnet.thread_pool.threads.count`	Number of thread pool threads that currently exist.
Lock Acquiring for the Last Hour	`process.runtime.dotnet.monitor.lock_contention.count`	

Process Metrics

Process metrics contain information on OS process performance.

Panel	Collected Metric	Description
Thread Count	`process.thread.count`	Number of process threads.

The amount of physical memory in use	`process.memory.usage`	The amount of physical memory utilized by the process.
The amount of committed virtual memory	`process.memory.virtual`	The amount of virtual memory allocated to the process.

Veeam Service Provider Console Custom Plugins Dashboard

Veeam Service Provider Console Custom Plugins Dashboard allows you to monitor resource usage and request handling by custom plugins that run on the Veeam Service Provider Console server side. Data collection is performed every 2 minutes or less.

All plugin metrics are grouped by the `pluginId` tag. Optionally you can group metrics by the `pluginPackageId` tag. The dashboard includes the drop-down filters for both tags.

The dashboard includes the following panels:

Panel	Collected Metric	Description
Plugins CPU usage	`vspc.plugins.cpu_usage`	Average CPU utilization across all cores for each plugin, in percents.
Plugins memory usage	`vspc.plugins.memory_usage`	Memory usage for each plugin.
Handled requests	`vspc.plugins.requests_handled`	Number of requests handled by each plugin.
Request processing time	`vspc.plugins.request_time_bucket`	Distribution bucket for plugin request processing time.

Veeam Service Provider Console Backend Statistic Dashboard

Veeam Service Provider Console Backend Statistic Dashboard allows you to monitor Veeam Service Provider Console infrastructure entities and their dynamics. Initial data collection is perforem in approximately 5 minutes after the service restart. After that, data is updated every 24 hours.

The dashboard includes the following panels.

Alarms

Panel	Collected Metric	Description
Enabled alarms	`vspc.statistic.alarms_total_active_count`	Number of enabled alarms.
Triggered alarms	`vspc.statistic.alarms_total_triggered_count`	Number of triggered alarms.
Configured alarm actions	`vspc.statistic.alarms_action_script_count`	Number of configured alarm actions.

Backup Agent Jobs

Panel	Collected Metric	Description
Microsoft Windows backup agents	`vspc.statistic.backup_agent_windows_total_count`	Total number of Microsoft Windows backup agents.
Linux backup agents	`vspc.statistic.backup_agent_linux_total_count`	Total number of Linux backup agents.
Mac OS backup agents	`vspc.statistic.backup_agent_mac_total_count`	Total number of Mac OS backup agents.

Backup agents	`vspc.statistic.backup_agent_total_count`	Total number of all backup agents.
Microsoft Windows backup agent jobs	`vspc.statistic.backup_agent_windows_jobs_total_count`	Total number of backup jobs processed by Microsoft Windows backup agents.
Linux backup agent jobs	`vspc.statistic.backup_agent_linux_jobs_total_count`	Total number of backup jobs processed by Linux agents.
Mac OS backup agent jobs	`vspc.statistic.backup_agent_mac_jobs_total_count`	Total number of backup jobs processed by Mac OS agents.
Backup agent jobs	`vspc.statistic.backup_agent_jobs_total_count`	Total number of backup agent jobs.
Microsoft Windows backup agents by company	`vspc.statistic.backup_agent_windows_company_count`	Number of Microsoft Windows backup agents for each company. Grouped by the `CompanyName` tag.
Linux backup agents by company	`vspc.statistic.backup_agent_linux_company_count`	Number of Linux backup agents for each company. Grouped by the `CompanyName` tag.
Mac OS backup agents by company	`vspc.statistic.backup_agent_mac_company_count`	Number of Mac OS backup agents for each company. Grouped by the `CompanyName` tag.
Backup agents by company	`vspc.statistic.backup_agent_company_count`	Number of backup agents for each company. Grouped by the `CompanyName` tag.
Microsoft Windows backup		Number of Microsoft Windows backup agent jobs for each

Microsoft Windows backup agent jobs by company	`vspc.statistic.backup_agent_windows_jobs_company_count`	Number of Microsoft Windows backup agent jobs for each company. Grouped by the `CompanyName` tag.
Linux backup agent jobs by company	`vspc.statistic.backup_agent_linux_jobs_company_count`	Number of Linux backup agent jobs for each company. Grouped by the `CompanyName` tag.
Mac OS backup agent jobs by company	`vspc.statistic.backup_agent_mac_jobs_company_count`	Number of Mac OS backup agent jobs for each company. Grouped by the `CompanyName` tag.
Backup agent jobs by company	`vspc.statistic.backup_agent_jobs_company_count`	Number of backup agent jobs for each company. Grouped by the `CompanyName` tag.

Veeam Cloud Connect

Panel	Collected Metric	Description
Cloud Connect servers	`vspc.statistic.cloud_connect_servers_count`	Total number of Veeam Cloud Connect servers.
Tenants per Cloud Connect server	`vspc.statistic.cloud_connect_tenants_count`	Number of tenants for each Veeam Cloud Connect server. Grouped by the `BackupServerName` tag.
Sub-tenants per Cloud Connect server	`vspc.statistic.cloud_connect_subtenants_count`	Number of sub-tenants for each Veeam Cloud Connect server. Grouped by the `BackupServerName` tag.
Cloud Director instances per VCC server	`vspc.statistic.cloud_connect_vcloud_directors_count`	Number of VMware Cloud Director instances for each Veeam Cloud Connect server. Grouped by the `BackupServerName` tag.

Cloud gateways per Cloud Connect server	`vspc.statistic.cloud_connect_cloud_gateways_count`	Number of cloud gateways for each Veeam Cloud Connect server. Grouped by the `BackupServerName` tag.
WAN accelerators per Cloud Connect server	`vspc.statistic.cloud_connect_wan_accelerators_count`	Number of WAN accelerators for each Veeam Cloud Connect server. Grouped by the `BackupServerName` tag.
Hardware plans per Cloud Connect server	`vspc.statistic.cloud_connect_hardware_plans_count`	Number of hardware plans for each Veeam Cloud Connect server. Grouped by the `BackupServerName` tag.
Cloud Connect servers by version	`vspc.statistic.cloud_connect_servers_version_count`	Number of Veeam Cloud Connect servers for each server version. Grouped by the `Version` tag.

Companies

Panel	Collected Metric	Description
Companies	`vspc.statistic.companies_total_count`	Total number of companies.
Companies by tenant type	`vspc.statistic.companies_by_tenant_type_count`	Number of companies for each tenant type. Grouped by the `CompanyTenantType` tag.
Companies by geo location	`vspc.statistic.companies_by_geo_location_defined_count`	Number of companies for each location. Grouped by the `CompanyGeolocationEnabled` tag.
Locations by company	`vspc.statistic.companies_location_count`	Number of backup policies for each company. Grouped by the `CompanyName` tag.

Backup policies by company	`vspc.statistic.companies_backup_policies_count`	Total number of backup jobs processed by Windows agents. Grouped by the `CompanyName` tag.
Users by company	`vspc.statistic.companies_users_count`	Number of users for each company. Grouped by the `CompanyName` tag.
Discovery rules by company	`vspc.statistic.companies_managed_vms_count`	Number of discovery rules for each company. Grouped by the `CompanyName` tag.
Managed VMs by company	`vspc.statistic.backup_agent_jobs_total_count`	Number of managed VMs for each company. Grouped by the `CompanyName` tag.
Backup agents in workstation mode by company	`vspc.statistic.companies_managed_agents_workstations_count`	Number of workstation agents for each company. Grouped by the `CompanyName` tag.
Backup agents in server mode by company	`vspc.statistic.companies_managed_agents_servers_count`	Number of server agents for each company. Grouped by the `CompanyName` tag.
Hosted VBR jobs by company	`vspc.statistic.companies_hosted_vbr_jobs_count`	Number of hosted Veeam Backup & Replication jobs for each company. Grouped by the `CompanyName` tag.
Assigned hosted vCenter Server tags	`vspc.statistic.companies_hosted_vbr_tags_assigned_count`	Number of assigned hosted vCenter Server tags for each company. Grouped by the `CompanyName` tag.
Assigned hosted Cloud Directors per company	`vspc.statistic.companies_hosted_vbr_vcd_assigned_count`	Number of assigned hosted Cloud Directors for each company. Grouped by the `CompanyName` tag.

ConnectWise Manage Integrations

Panel	Collected Metric	Description
Configured ConnectWise Manage integrations	`vspc.statistic.integrations_cwm_count`	Number of configured ConnectWise Manage integrations.

Discovery

Panel	Collected Metric	Description
Discovered rules by company	`vspc.statistic.discovery_rules_count`	Number of discovered rules for each company. Grouped by the `CompanyName` tag.

Organization Count

Panel	Collected Metric	Description
Active companies	`vspc.statistic.companies_active_count`	Number of active companies.
Active resellers	`vspc.statistic.resellers_active_count`	Number of active resellers.

Resellers

Panel	Collected Metric	Description
Resellers	`vspc.statistic.resellers_total_count`	Total number of resellers.

Companies per reseller	`vspc.statistic.resellers_company_count`	Number of companies for each reseller. Grouped by the `ResellerName` tag.
Users per reseller	`vspc.statistic.resellers_users_count`	Number of users for each reseller. Grouped by the `ResellerName` tag.
Resellers by status	`vspc.statistic.resellers_count_by_status`	Number of resellers for each reseller status. Grouped by the `Status` tag.

Management Agents

Panel	Collected Metric	Description
Management agents	`vspc.statistic.management_agents_count`	Total number of management agents.
Management agents by guest OS	`vspc.statistic.management_agents_by_os_count`	Number of management agents for each guest OS. Grouped by the `OperatingSystem` tag.
Management agents by version	`vspc.statistic.management_agents_version_count`	Number of management agents for each version. Grouped by the `Version` tag.

Veeam Backup & Replication Servers

Panel	Collected Metric	Description
Remote Veeam Backup &	`vspc.statistic.vbr_servers`	Number of remote Veeam Backup & Replication servers.

Replication servers	`client_count`	
Hosted Veeam Backup & Replication servers	`vspc.statistic.vbr_servers_hosted_count`	Number of hosted Veeam Backup & Replication servers.
Jobs per Veeam Backup & Replication server	`vspc.statistic.vbr_servers_job_count`	Number of jobs for each Veeam Backup & Replication server. Grouped by the `BackupServerName` tag.
Proxies per Veeam Backup & Replication server	`vspc.statistic.vbr_servers_proxies_count`	Number of proxies for each Veeam Backup & Replication server. Grouped by the `BackupServerName` tag.
Repositories per Veeam Backup & Replication server	`vspc.statistic.vbr_servers_repositories_count`	Number of repositories for each Veeam Backup & Replication server. Grouped by the `BackupServerName` tag.
Remote Veeam Backup & Replication servers by version	`vspc.statistic.backup_servers_client_version_count`	Number of remote Veeam Backup & Replication servers for each version. Grouped by the `Version` tag.
Hosted Veeam Backup & Replication servers by version	`vspc.statistic.backup_servers_hosted_version_count`	Number of hosted Veeam Backup & Replication servers for each version. Grouped by the `Version` tag.

Backup Agents

Panel	Collected Metric	Description
Microsoft Windows backup agents by version	`vspc.statistic.backup_agent_windows_version_count`	Number of Microsoft Windows backup agents for each version. Grouped by the `Version` tag.
Linux backup agents by	`vspc.statistic.backup_agent_li`	Number of Linux backup agents for each version. Grouped

version	`nux_version_count`	by the `Version` tag.
Mac backup agents by version	`vspc.statistic.backup_agent_mac_version_count`	Number of Mac backup agents for each version. Grouped by the `Version` tag.
Hosted Microsoft Windows agents	`vspc.statistic.backup_agent_windows_hosted_count`	Number of hosted Microsoft Windows agents.
Hosted Linux agents	`vspc.statistic.backup_agent_linux_hosted_count`	Number of hosted Linux agents.
Hosted Mac agents	`vspc.statistic.backup_agent_mac_hosted_count`	Number of hosted Mac agents.

Protected Data

Panel	Collected Metric	Description
Protected VMs	`vspc.statistic.protected_data_vm_total_count`	Number of protected virtual machines.
Protected VMs (Hyper-V)	`vspc.statistic.protected_data_vm_hyperv_count`	Number of protected Microsoft Hyper-V virtual machines.
Protected VMs (vSphere)	`vspc.statistic.protected_data_vm_vsphere_count`	Number of protected VMware vSphere virtual machines.
Protected VMs (AHV)	`vspc.statistic.protected_data_vm_ahv_cou`	Number of protected AHV virtual machines.

	`nt`	
Protected VMs (AWS)	`vspc.statistic.protected_data_vm_aws_count`	Number of protected AWS virtual machines.
Protected VMs (Azure)	`vspc.statistic.protected_data_vm_azure_count`	Number of protected Azure virtual machines.
Protected VMs (Google Cloud)	`vspc.statistic.protected_data_vm_googlecloud_count`	Number of protected Google Cloud virtual machines.
Protected servers managed by VSPC	`vspc.statistic.protected_data_computers_managed_by_vspc_servers_count`	Number of protected servers managed by Veeam Service Provider Console.
Protected workstations managed by VSPC	`vspc.statistic.protected_data_computers_managed_by_vspc_workstations_count`	Number of protected workstations managed by Veeam Service Provider Console.
Protected servers managed by VBR	`vspc.statistic.protected_data_computers_managed_by_vbr_servers_count`	Number of protected servers managed by Veeam Backup & Replication.
Protected workstations managed by VBR	`vspc.statistic.protected_data_computers_managed_by_vbr_workstations_count`	Number of protected workstations managed by Veeam Backup & Replication.
Protected file shares	`vspc.statistic.protected_data_file_shares_count`	Number of protected file shares.
Data size of protected file shares	`vspc.statistic.protected_data_file_shares_amount_data_size`	Total data size of protected file shares.

Protected AWS EFS shares	`vspc.statistic.protected_data_aws_efs_count`	Number of protected AWS EFS shares.
Protected AWS FSx shares	`vspc.statistic.protected_data_aws_fsx_count`	Number of protected AWS FSx shares.
Protected Azure file shares	`vspc.statistic.protected_data_azure_count`	Number of protected Azure file shares.
Protected object storages	`vspc.statistic.protected_data_object_storages_count`	Number of protected object storages.
Data size of protected object storages	`vspc.statistic.protected_data_object_storages_data_size`	Total data size of protected object storages.
Protected AWS databases	`vspc.statistic.protected_data_database_aws_count`	Number of protected AWS databases.
Protected Azure databases	`vspc.statistic.protected_data_database_azure_count`	Number of protected Azure databases.
Protected Google Cloud databases	`vspc.statistic.protected_data_database_google_cloud_count`	Number of protected Google Cloud databases.
Protected Google Spanner instances	`vspc.statistic.protected_data_database_google_spanner_count`	Number of protected Google Spanner instances.
Protected Azure Cosmos DB	`vspc.statistic.protected_data_database_a`	Number of protected Azure Cosmos DB

Protected Azure Cosmos DB instances	`vspc.statistic.protected_data_database_azure_cosmos_count`	Number of protected Azure Cosmos DB instances.
Protected AWS DynamoDB tables	`vspc.statistic.protected_data_database_aws_dynamodb_count`	Number of protected AWS DynamoDB tables.
Protected AWS Redshift clusters	`vspc.statistic.protected_data_database_aws_redshift_count`	Number of protected AWS Redshift clusters.
Protected AWS networks	`vspc.statistic.protected_data_network_aws_count`	Number of protected AWS networks.
Protected Azure networks	`vspc.statistic.protected_data_network_azure_count`	Number of protected Azure networks.
Protected Microsoft 365 users	`vspc.statistic.protected_data_microsoft365_users_count`	Number of protected Microsoft 365 users.
Protected Microsoft 365 groups	`vspc.statistic.protected_data_microsoft365_groups_count`	Number of protected Microsoft 365 groups.
Protected Microsoft 365 sites	`vspc.statistic.protected_data_microsoft365_sites_count`	Number of protected Microsoft 365 sites.
Protected Microsoft 365 teams	`vspc.statistic.protected_data_microsoft365_teams_count`	Number of protected Microsoft 365 teams.

License Information

Panel	Collected Metric	Description
Licensed units	`vspc.statistic.licensing_licensed_units`	Number of license units.
Used license units	`vspc.statistic.licensing_used_units`	Number of used license units.
New license units	`vspc.statistic.licensing_new_units`	Number of new license units.

SQL Server

Panel	Collected Metric	Description
Database table size (MB)	`vspc.statistic.database_sql_table_total_size`	Total database table size. Grouped by the `TableName` tag.
Used database table size (MB)	`vspc.statistic.database_sql_table_used_size`	Used database table size. Grouped by the `TableName` tag.

Single Sign-On

Panel	Collected Metric	Description
Configured identity providers	`vspc.statistic.identity_providers_count`	Number of configured identity providers.

Authentication

Panel	Collected Metric	Description

Panel	Collected Metric	Description
User authentication attempts last 30 days	`vspc.statistic.users_authentications_last_30_days_count`	Number of user authentication attempts during the last 30 days.

Time Formats

In Veeam Service Provider Console REST API, there are several date and time formats present:

- `date-time` - has the standard `yyyy-MM-ddTHH:mm:ss.fffffffK` structure according to RFC 3339. For example, `2023-01-16T16:30:09.4952609+01:00`.
- `time-of-day` - has the HH:MM structure. For example, `10:30`.
- `time` - has the HH:MM:SS structure. For example, `10:30:03`.

REST API Client

REST API Client is an application that provides optimized authentication, error processing, throttling management and PATCH operation handling. The client is a NuGet package for .NET platform located in the `REST Client` folder of the installation disk image.

For details on the client capabilities, see the `readme.md` file located in the package.

Evaluation with Swagger UI

To start working with the Veeam Service Provider Console REST API, you can use any client application that supports the HTTPS protocol.

TIP

You can generate your own client using Swagger Generator. For more information, see Online Generators.

The Veeam Service Provider Console REST API is additionally available through Swagger UI, a tool that you can use to evaluate and explore capabilities of the Veeam Service Provider Console REST API. Swagger UI visually presents API specification files and allows you to work with resources.

All resources and methods are expandable. When you expand a method, you get a full description of available parameters and an automatically generated example. You can also send requests and see response messages.

To access Swagger UI for the Veeam Service Provider Console REST API:

1. In the Internet browser address bar, enter the Veeam Service Provider Console REST API URL in the following format: `https://<hostname>:<port>/api`.
2. On the landing page, click **SwaggerUI**.

Alternatively, in the Internet browser address bar, you can enter the Veeam Service Provider Console REST API URL in the following format: `https://<hostname>:<port>/api/swagger`.

By default, all resources are visible. To view only those resources that are available to users with a specific user role, open a drop down list in the upper right corner of the window and select the role.

To download the REST API specification with role specific resources, select the role and click the link under the **VSPC REST** header.

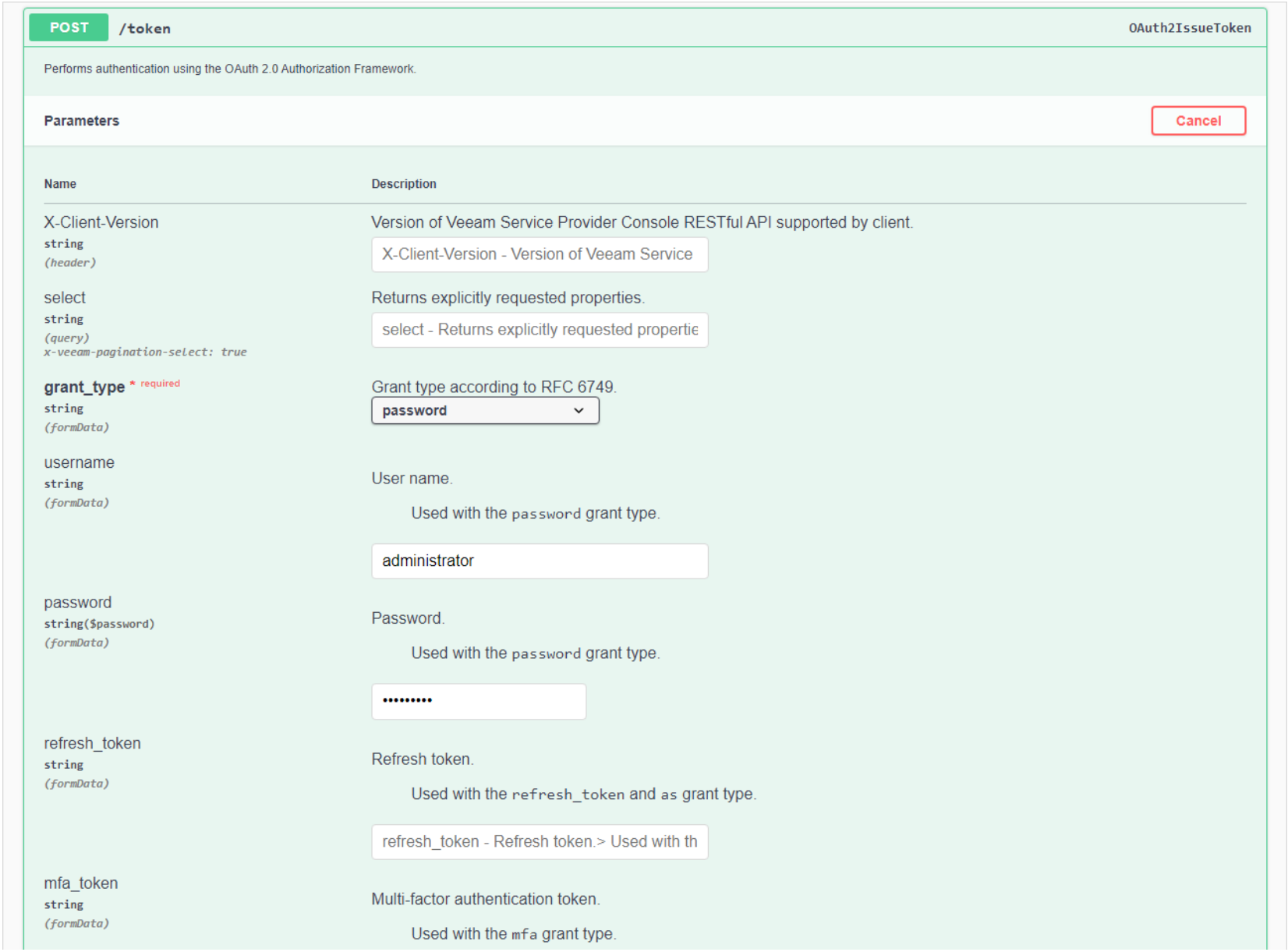
Getting Authorization Tokens

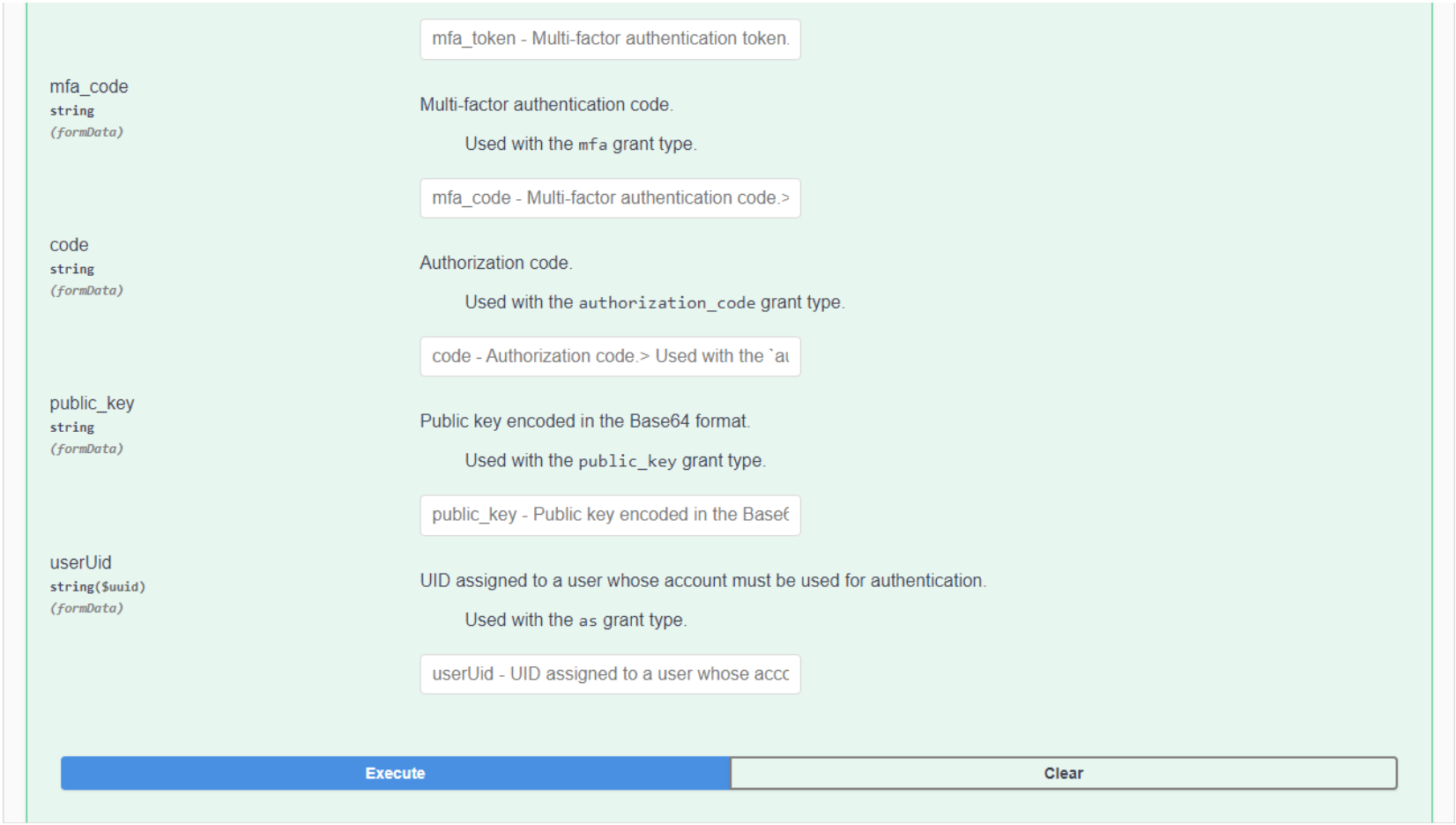
The Veeam Service Provider Console REST API authorization with Swagger UI involves the following procedures:

1. Obtain an access token:
    a. On the **VSPC REST** page, expand the **Authentication** resource and click **POST /token**.

b. Click the **Try it out** button.

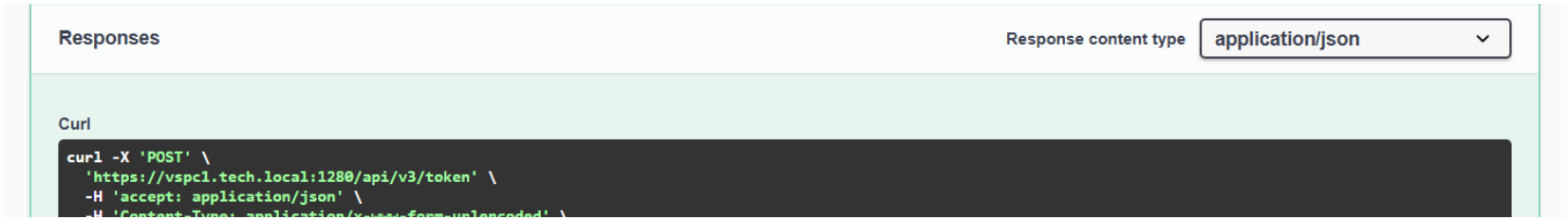
c. In the **username** and **password** fields, specify the credentials of a user with the privileges of a user that has access to the Veeam Service Provider Console REST API.

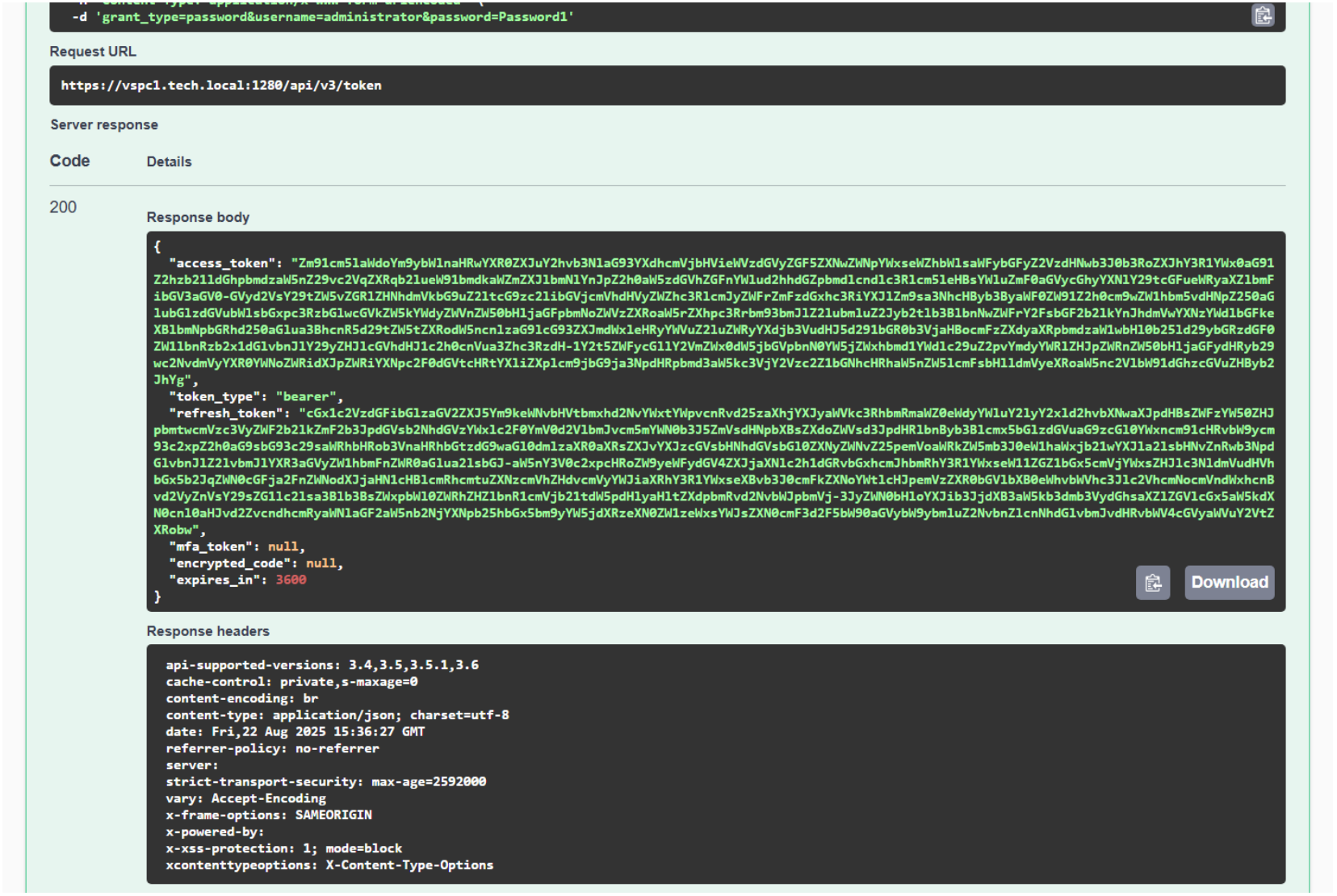




d. Click **Execute**.

Wait for the response from the server. A successfully completed operation returns the *200 OK* response code and an access token in the response body.



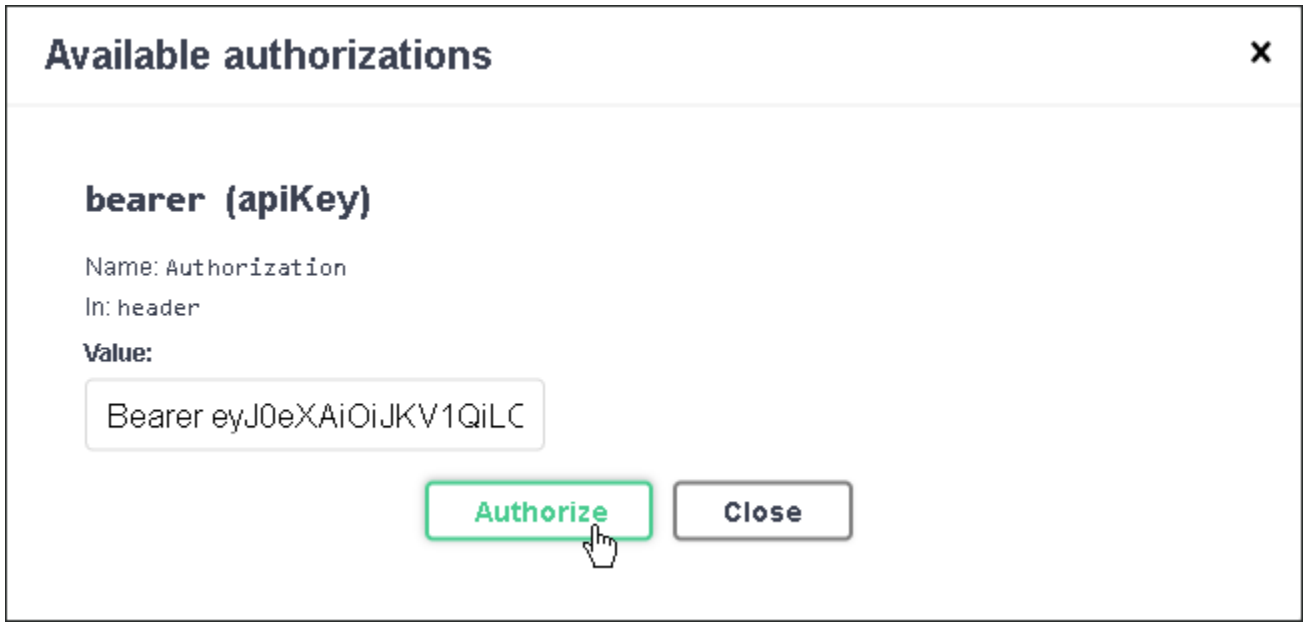


e. Copy the access token.

f. Click the **Authorize** button. The **Available authorizations** window will open.

g. In the **Value** field, type *Bearer* and the space character, then paste the access token.

h. Click **Authorize**.



2. When the access token expires, use the refresh token:

   a. On the **VSPC REST** page, expand the **Authentication** resource and click **POST /token**.

   b. Click **Try it out**.

   c. In the **grant_type** drop-down list, select **refresh_token**.

   d. In the **refresh_token** field, insert the refresh token saved locally.

   e. Click **Execute**.

   Wait for the response from the server. A successfully completed operation returns the *200 OK* response code and a new pair of tokens in the response body.

3. When you finish working with the Veeam Service Provider Console REST API:

   a. Click the **Authorize** button. The **Available authorizations** window will open.

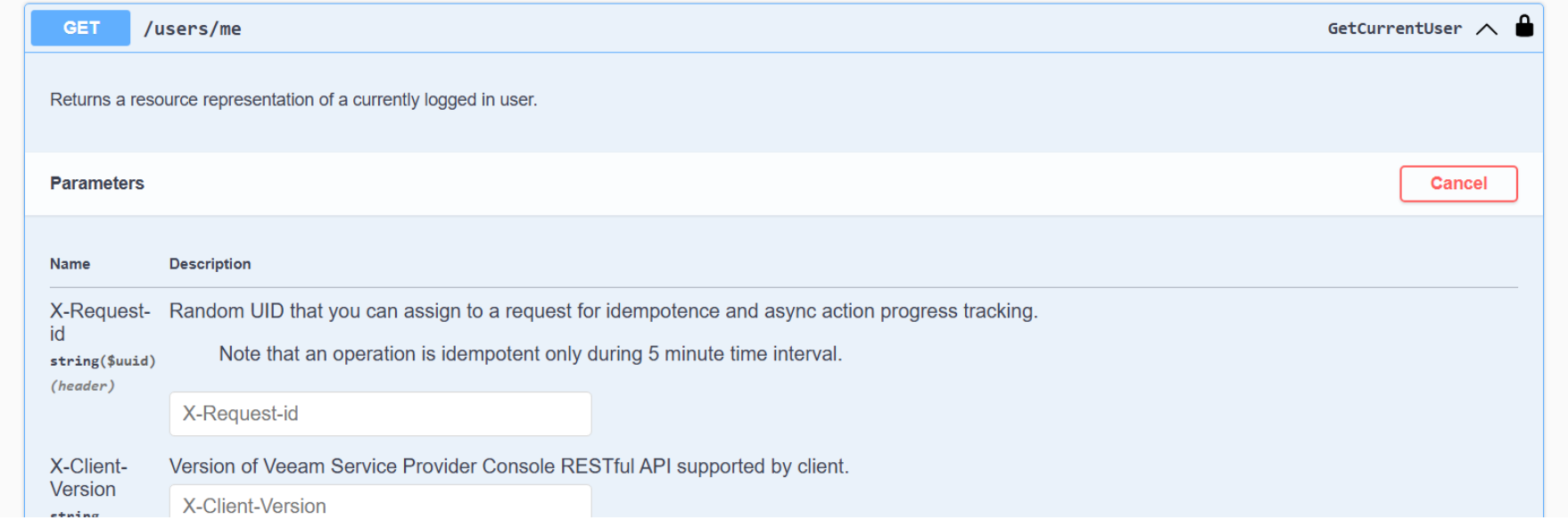
b. Click the **Logout** button.

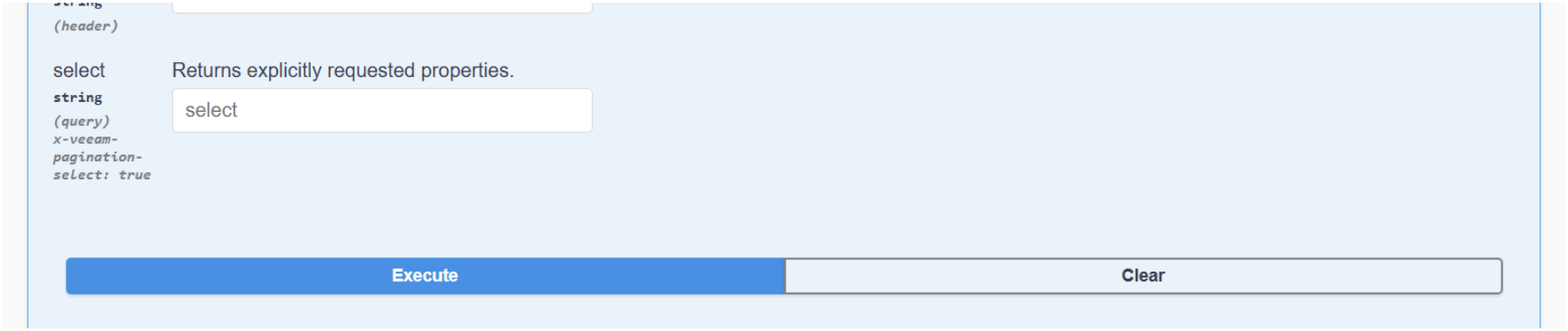
Sending Requests

After you get and validate an access token, you can send HTTPS requests to the Veeam Service Provider Console REST API collections and resources in Swagger UI.

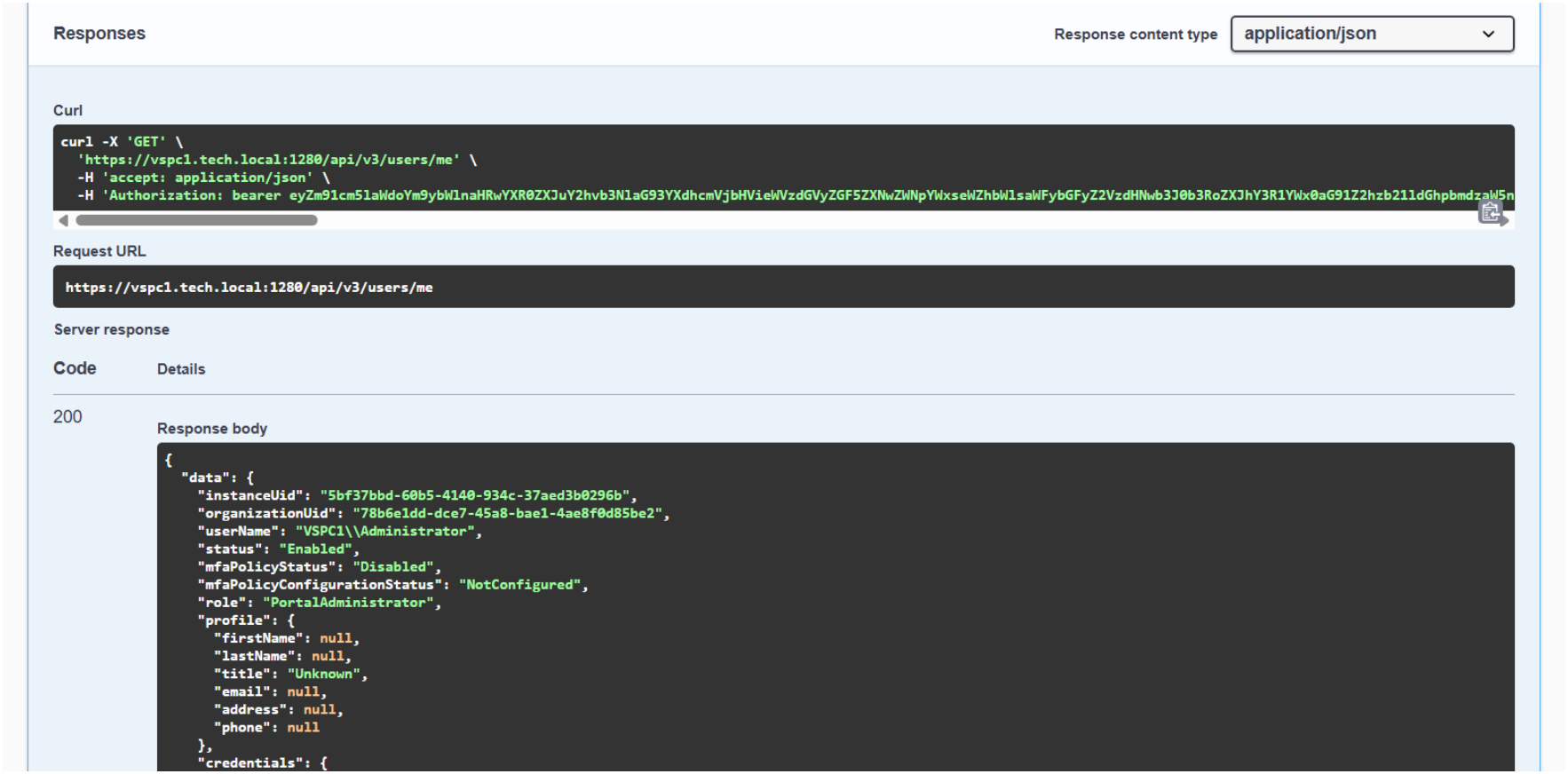
To send a request:

1. On the **VSPC REST** page, expand a resource to which you want to perform an operation.
2. In the list of request methods, click the required method.
3. Click **Try it out**.
4. In the expanded method window, select a response content type and enter parameter values.
5. Click **Execute**.





Swagger UI returns a response body as well as a response code and response headers. Additionally, Swagger UI generates a cURL command and a URL for your request.



```
      "userName": "VSPC1\\Administrator",
      "password": null
    }
  }
}
```


Download

Response headers


```
api-supported-versions: 3.4,3.5,3.5.1,3.6
cache-control: private,s-maxage=0
content-encoding: br
content-type: application/json; charset=utf-8
date: Fri,22 Aug 2025 16:27:17 GMT
referrer-policy: no-referrer
server:
strict-transport-security: max-age=2592000
vary: Accept-Encoding
x-frame-options: SAMEORIGIN
x-powered-by:
x-xss-protection: 1; mode=block
xcontenttypeoptions: X-Content-Type-Options
```