IBM Spectrum Protect Plus
10.1.12

REST API Guide

Note:

Before you use this information and the product it supports, read the information in “Notices” on page 249.

This edition applies to version 10, release 1, modification 12 of IBM Spectrum® Protect Plus (product number 5737-F11) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this publication

IBM Spectrum Protect Plus is a data protection solution that provides recovery, replication, retention, and reuse for virtual machines, databases, and containers in hybrid multi-cloud environments. It unlocks the value of your data by improving the quality and speed of development, testing, and analytics. Data retention, data compliance, and disaster recovery are achieved by copying or archiving data to both on-premises and cloud-based object storage and IBM Spectrum Protect, including support for the physical and virtual tape storage.

IBM Spectrum Protect Plus offers an application programming interface (API) based on Hypertext Transfer Protocol (HTTP) and Representational State Transfer (REST) architecture. APIs using REST architecture and HTTP are often called REST APIs or RESTful APIs.

The REST API and related components comply with the industry standards for HTTP and REST architecture. Here is a list of documents defining those standards recommended by Internet Engineering Task Force (IETF) as Request for Comments (RFCs):

- Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, IETF RFC 7231.
- The JavaScript Object Notation (JSON) Data Interchange Format, IETF RFC 8259.
- Common Format and MIME Type for Comma-Separated Values (CSV) Files, IETF RFC 4180.

This publication provides information about the IBM Spectrum Protect Plus API based on the RESTful framework. Use the RESTful API to connect your applications to product components in order to query information about objects and to run basic operations by using HTTP protocols and the principles of RESTful API. This API uses RESTful architecture designed to interact with web-based applications in a simplified way, by using four basic HTTP methods: GET, POST, PUT, and DELETE.

Who should read this publication

This publication is intended for developers who want to use the RESTful API to connect to IBM Spectrum Protect Plus components to query information about objects and to run basic operations.

In this publication, it is assumed that you have an understanding of the components and features of IBM Spectrum Protect Plus as described in the *IBM Spectrum Protect Plus Installation and User's Guide*.

Publications

The IBM Spectrum Protect product family includes IBM Spectrum Protect Plus, IBM Spectrum Protect for Virtual Environments, IBM Spectrum Protect for Databases, and several other storage management products from IBM®.

To view IBM product documentation, see IBM Documentation.

Chapter 1. The IBM Spectrum Protect Plus REST API

A REST API is provided as part of IBM Spectrum Protect Plus.

This document is a good starting reference for those interested in automating basics tasks in IBM Spectrum Protect Plus. You can invoke actions by using the IBM Spectrum Protect Plus REST API.

In addition to this guide, sample source code is available through the `sppclient` module. The `sppclient` module simplifies interaction with the IBM Spectrum Protect Plus REST API. This module can be used for common tasks on the IBM Spectrum Protect Plus appliance or used for automation of testing. For more information on the `sppclient` module, see https://github.com/IBM/sppclient.

There is also the SPPMon project. This open-source project is initiated by the IBM Spectrum Protect Plus development team and provides a monitoring system for IBM Spectrum Protect Plus with multiple options for day to day monitoring of a data protection environment. For more information on the SPPMon project, see https://github.com/IBM/spectrum-protect-sppmon.

Structure and methodology of the REST API

In IBM Spectrum Protect Plus, you use a **session ID** that is associated with a specific user account to send an HTTP request. After you request a session ID for a user account by providing the user name and password for the account, you must ensure that every HTTP request includes a valid session ID.

Before you begin

Ensure that you have the required permissions for any actions that you are planning. Permission settings for user accounts are managed by role-based access control. For more information, see the IBM Spectrum Protect Plus User's and Installation Guide.

Methodology

Before you develop an application by using the IBM Spectrum Protect Plus REST API, consider using this methodology:

- Run the procedure that you plan to develop by accessing IBM Spectrum Protect Plus through a web browser. Use several example instances and scenarios.
- Generalize and design the procedure with the API.

To use the suggested methodology, complete the following steps:

- Open IBM Spectrum Protect Plus. Open the login page and enter the username and the password of a specific user.

  If the credentials are valid, you access the Dashboard page as the user. In the back end, your browser receives 32 hexadecimals called session ID from IBM Spectrum Protect Plus and stores the information temporarily. Every session ID is associated with that user. The session ID is valid for 45 minutes. If you navigate to another page in IBM Spectrum Protect Plus, you do not have to enter the username and the password because the session ID in your browser is used automatically. Keep the page open and do nothing for 45 minutes and you will log out of the session automatically.
- Access the IBM Spectrum Protect Plus REST API. If you have a valid session ID, use it; otherwise, request a new one by sending an HTTP request command with a username and a password, as described in

  “Creating a session ID” on page 15

  .

You will receive an HTTP response that contains a session ID associated with that user. Use this session ID in your HTTP requests to invoke other actions such as running inventory jobs for hypervisors.

Response and request format: JSON

The IBM Spectrum Protect Plus REST API uses JavaScript Object Notation (JSON) for HTTP requests and responses.

Structure of an HTTP response message

The REST API follows a consistent structure for response messages. For example, you might send the following request to IBM Spectrum Protect Plus by using an IP address of 10.0.0.100 to get site information.

```
GET     https://10.0.0.100/api/site
```

You receive the following response with the HTTP status code of 200 (OK).

```
{
    "links": {
        "self": {
            "rel": "self",
            "href": "https://10.0.0.100/api/site",
            "hreflang": null,
            "media": null,
            "title": null,
            "type": null,
            "deprecation": null
        },
        "create": {
            "rel": "create",
            "href": "https://10.0.0.100/api/site",
            "hreflang": null,
            "media": null,
            "title": null,
            "type": null,
            "deprecation": null
        }
    },
    "total": 3,
    "sites": [
        {
            "links": {
                "self": {
                    "rel": "self",
                    "href": "https://10.0.0.100/api/site/1000",
                    "hreflang": null,
                    "media": null,
                    "title": null,
                    "type": null,
                    "deprecation": null
                },
                "up": {
                    "rel": "up",
                    "href": "https://10.0.0.100/api/site",
                    "hreflang": null,
                    "media": null,
                    "title": null,
                    "type": null,
                    "deprecation": null
                },
                "edit": {
                    "rel": "update",
                    ...
                },
                "delete": {
                ...
                },
                "usedby": {
                    "rel": "related",
                    "href": "http://localhost:8082/api/endeavour/association/
resource/site/1000?action=listUsingResources",
                    ...
                },
                "resourceprovider": {
                    "rel": "related",
                    "href": "https://10.0.0.100/api/site/1000/resourceprovider",
```

```
                    ...
                }
            },
            "id": "1000",
            "name": "Primary",
            "description": "Primary Site",
            "defaultSite": false,
            "throttles": null,
            "demo": false,
            "rbacPath": "root:0/site:0/site:1000"
        },
        {...},
        {...}
    ]
}
```

The structure above can be generalized as below:

```
{
    "links":        <HATEOAS>,
    "total":        <objectCount>,
    "<objectType>": [
        {
            "links":          <HATEOAS>,
            "id":             <objectId>,
            "name":           <objectName>,
            "<requiredKey1>": <requiredValue1>,
            "<requiredKey2>": <requiredValue2>,
            ...,
            "<optionalKey1>": <optionalValue1>,
            "<optionalKey2>": <optionalValue2>,
            ...
        },
        {
            "links":          <HATEOAS>,
            "id":             <objectId>,
            "name":           <objectName>,
            "<requiredKey1>": <requiredValue1>,
            "<requiredKey2>": <requiredValue2>,
            ...,
            "<optionalKey1>": <optionalValue1>,
            "<optionalKey2>": <optionalValue2>,
            ...
        },
        <object3>,
        <object4>
        ...
    ]
}
```

HATEOAS and URI representation

Hypermedia as the Engine of Application State (HATEOAS) is a common methodology to provide users with interfaces of applications using REST architecture. Here is an example in IBM Spectrum Protect Plus. Recall the example response in the previous section and see what actions are represented as URIs based on HATEOAS.

The IBM Spectrum Protect Plus REST API has operation parameters that can transform a JSON object in a response. For details, follow the instructions in

“Operation parameters for JSON object output” on page 243.

Getting started

You can develop an application that interacts with IBM Spectrum Protect Plus via the REST API by using various platforms, programming languages, scripting languages, and development tools.

Developing a client application by using the curl tool

curl is a cross-platform command-line data transfer tool that works well with REST transactions. Mozilla Firefox and Google Chrome can record the browser's HTTP requests and transform the requests into a curl command format. In a shell session, you can invoke the `curl` command and combine it with other text-processing commands such as `grep`. Although working with shell sessions is not suitable for processing large JSON records, it is a good approach to capture a browser's HTTP request in the curl format and modify it to generalize the structure before you actually code them in the production environment such as Python and Java.

This section describes how you can retrieve any actions on the IBM Spectrum Protect Plus web user interface on Mozilla Firefox or Google Chrome, capture their HTTP requests in the curl format, and then reuse them from your shell session. curl supports many platforms, for example:

- Bash on Linux and Microsoft Windows Subsystems for Linux (WSL) 1
- Z Shell on Apple macOS series
- PowerShell Core and Microsoft Windows PowerShell and Command shell

For more information about curl, see the following documentation:

- curl.1 the manpage
- Everything curl, ISBN 978-91-639-6501-2

Capturing HTTP requests on a web browser

IBM Spectrum Protect Plus offers two user interfaces: the web user interface and the REST API. For scripting, you can take the following approach:

- From a supported browser, take the action that you want to script and determine the simplest path.
- Complete the action with multiple scenarios while capturing HTTP transactions.
- Review the HTTP transactions that you captured.
- Script the HTTP requests and handle the expected responses.

The supported browsers (Mozilla Firefox, Google Chrome, and Microsoft Edge) have a developer tool that can be used to capture HTTP transactions.

In **Mozilla Firefox**, open **Firefox Developer Tools**. Click the **Network** tab. Click the **Pause/Resume recording network log** button to capture HTTP requests. Take the action that you want to script in the IBM Spectrum Protect Plus web user interface. Click the button again to stop capturing further HTTP requests. Find the relevant HTTP request in the list. Right-click the request, click **Copy** > **Copy as cURL (POSIX)**. Ensure that the curl command that you have copied works as you expect. Paste the command into a text editor and examine it.

In **Google Chrome**, you can capture HTTP requests in a similar way as in the example with Mozilla Firefox. Open **DevTools**. Click the **Network** tab. Click **Record network log**. In the IBM Spectrum Protect Plus web user interface, take the actions that you want to script. Click **Pause recording network log**. Find the relevant HTTP request in the list. Right-click the request, click **Copy** > **Copy as cUrl (bash)**. Ensure that the curl command you copied works as you expect. Paste the command into a text editor and examine it.

Before you begin

To review the web browsers that are supported for IBM Spectrum Protect Plus, see IBM Spectrum Protect Plus System Requirements.

In **Mozilla Firefox**, open **Firefox Developer Tools**. Click the **Network** tab. Click the **Pause/Resume recording network log** button to capture HTTP requests. Take the action that you want to script in the IBM Spectrum Protect Plus web user interface. Click the button again to stop capturing further HTTP requests. Find the relevant HTTP request in the list. Right-click the request, click **Copy** > **Copy as cURL (POSIX)**. Ensure that the curl command that you have copied works as you expect. Paste the command into a text editor and examine it.

In **Google Chrome**, you can capture HTTP requests in a similar way as in the example with Mozilla Firefox. Open **DevTools**. Click the **Network** tab. Click **Record network log**. In the IBM Spectrum Protect Plus web user interface, take the actions that you want to script. Click **Pause recording network log**. Find the relevant HTTP request in the list. Right-click the request, click **Copy** > **Copy as cUrl (bash)**. Ensure that the curl command you copied works as you expect. Paste the command into a text editor and examine it.

Example: Start an inventory job for hypervisor instances

The following example shows how to capture an HTTP request to invoke an inventory job for hypervisors: VMware or Microsoft Hyper-V.

In your web browser, log in to IBM Spectrum Protect Plus (IPv4 address 10.0.0.100). In the navigation pane, click **Manage Protection** > **Hypervisors** > **VMware**. Turn on recording for a network log. Click **Run Inventory**. Turn off the recording mode.

In Mozilla Firefox 81 for Windows 10, for example, you will get the HTTP request and copy it into your system clipboard as a single curl command for POSIX:

```
curl 'https://10.0.0.100/api/endeavour/job/1004?action=start&actionname=start' \
  -H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:81.0) \
      Gecko/20100101 Firefox/81.0' \
  -H 'Accept: application/json' \
  -H 'Accept-Language: en-US,en;q=0.5' \
  --compressed \
  -H 'Content-Type: application/json'\
  -H 'X-endeavour-auditrequest: true' \
  -H 'X-Endeavour-Sessionid: 8b56b813936a4957be1db80ca8a8ea0c' \
  -H 'X-Endeavour-Locale: en-us' \
  -H 'Origin: https://10.0.0.100' \
  -H 'DNT: 1' \
  -H 'Connection: keep-alive' \
  -H 'Referer: https://10.0.0.100/' \
  -H 'Pragma: no-cache' \
  -H 'Cache-Control: no-cache' \
  --data ''
```

The curl command is equivalent to the following HTTP request:

Method: POST

URI: https://10.0.0.100/api/endeavour/job/1004

Parameters:

Key	Value
action	start
actionname	start

Tip: A Uniform Resource Identifier (URI) is typically encoded with ASCII characters and can include parameters. For information about the syntax of URIs, see the following document, including its errata: Uniform Resource Identifier (URI): Generic Syntax, IETF RFC 3986.

For instructions about decoding URIs, see the following topics:

- “Python REST API” on page 7
- “PowerShell REST API” on page 10

Data: None.

Header:

Key	Value
Accept	application/json
Content-Type	application/json

Key	Value
X-Endeavour-Sessionid	8b56b813936a4957be1db80ca8a8ea0c

Options:

- `--compressed`

For more information about the relevant REST API request command, follow the instructions in “Running an inventory job for virtualized systems” on page 64.

Developing a client application by using Postman

Postman is an API development tool that supports REST APIs. You can use Postman to design scripts that interact with the IBM Spectrum Protect Plus REST API. Postman is a desktop application that runs on the Microsoft Windows, Apple macOS, and Linux operating systems.

After you download Postman from the official site. As soon as you complete the setup and registration steps, you will see a workspace.

Example 1: Get an IBM Spectrum Protect Plus session ID

Assume that you want to get a session ID from the IBM Spectrum Protect Plus virtual appliance whose IPv4 address is 10.0.0.100. If you already have a valid session ID, reuse it and skip this step. For details, follow the instructions in

“Creating a session ID” on page 15

. Launch Postman and open a new workspace.

Set the HTTP method (left of the URL field) to **POST**. Enter the URL:

```
https://10.0.0.100/api/endeavour/session
```

Click **Params** to manage key-value pairs for the field in your HTTP request. You will see the params values being appended to the URL automatically. At this time, leave these fields blank.

Click **Auth** and enter appropriate values:

- **TYPE:** Basic Auth
- **Username:** (A valid username for the target IBM Spectrum Protect Plus application)
- **Password:** (The password for this user)

Click **Send**.

The response body and the HTTP status appear in the response pane. JSON records in the response body are formatted in color and with indentations (pretty-print format). You can collapse and expand nested JSON objects.

In this case, the key-value pair of `sessionid` should appear in the second line. Copy the value.

Example 2: Get site information

Set the HTTP method to **GET**. Enter the URL:

```
https://10.0.0.100/api/site
```

Click **Auth** and enter appropriate values:

- **TYPE:** API Key
- **Key:** X-Endeavour-Sessionid
- **Value:** (Paste the session ID)

- **Add to:** Header

Click **Send**.

Ensure that you get the HTTP status of 200 (OK). The response body shows the list of the sites in a JSON format.

Python REST API

By using the Python programming language, you can script a client application that interacts with the IBM Spectrum Protect Plus REST API. The examples in this section and throughout this guide use third-party library Requests for HTTP transactions.

Applicable Python releases: The provided examples are based on the following releases: Python series: Python 3.6, 3.7, 3.8, and 3.9.

Getting a session ID using Python

Assume that you want to work with an IBM Spectrum Protect Plus virtual appliance with the following configuration:

- **IPv4 address:** 10.0.0.100
- **Username:** Sarah
- **Password:** `MyPassw0rd!`
- **SSL verification:** False, ignore warnings.

The following Python libraries and values to access to the REST API.

```
import json
import requests

spp_ipv4 = '10.0.0.100'
spp_username = 'Sarah'
spp_password = 'MyPassw0rd!'
spp_verify = False              # Disable SSL.

# Supress HTTPS warnings when spp_verify is set to False.
from requests.packages.urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)
```

After the previous script runs, the following Python snippet is used to retrieve a session ID from the target IBM Spectrum Protect Plus application:

```
spp_session = requests.post('https://' + spp_ipv4 + '/api/endeavour/session',
    auth=(spp_username, spp_password),
    headers={
        'Accept':       'application/json',
        'Content-type': 'application/json'
    },
    verify=spp_verify
)

spp_session = json.loads(spp_session.text)    # Convert to JSON
spp_sessionid = spp_session['sessionid']

print(spp_sessionid)
```

```
 ee88d182812f49c98bbf9c819d69af07
```

You can use the obtained session ID for subsequent operations. For more information about session IDs, follow the instructions in

“Creating a session ID” on page 15.

Sending a request to the REST API by using Python

Assume that you append the following Python snippet to the previous script. The snippet sends an HTTP request to the REST API and starts an inventory job for virtualized systems, such as virtual machines hosted on vCenter Server. For more information about this operation, follow the instructions in

"Running an inventory job for virtualized systems" on page 64.

```
_params = {
    "action":    "start",
    "actionname": "start"
}

_data = ""

requests.post('https://' + spp_ipv4 + '/api/endeavour/job/1004,
    headers={
        'Accept':                'application/json',
        'Content-type':          'application/json',
        'X-Endeavour-Sessionid': spp_sessionid
    },
    params=_params, data=_data, verify=spp_verify
)
```

The structure of this POST command is similar to the command for getting a session ID except for the URI and the session ID in the header.

The IBM Spectrum Protect Plus REST API has a consistent structure. You can use Python scripts that are similar to this example to develop applications that send requests to the REST API. You can enhance those basic structures with many other methods and mechanisms that are available in Python and its third-party libraries.

Decoding URIs with parameters by using Python

You can decode URI strings with parameters by using Python snippets.

Assume that you captured the following HTTP request by using Mozilla Firefox:

```
curl 'https://10.0.0.100/api/hypervisor?from=hlo&pageSize=100&
   sort=%5B%7B%22property%22:%22name%22,%22direction%22:%22ASC%22%7D%5D&
   filter=%5B%7B%22property%22:%22type%22,%22value%22:%22vmware%22,
   %22op%22:%22=%22%7D%5D'
  -H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) ...'
...
```

A Python snippet that is similar to the following example can be used to decode the URI string:

```
import urllib.parse

uri_original = "https://10.0.0.100/api/hypervisor?" \
    + "from=hlo&" \
    + "pageSize=100&" \
    + "sort=%5B%7B%22property%22:%22name%22,%22direction%22:%22ASC%22%7D%5D&" \
    + "filter=%5B%7B%22property%22:%22type%22,%22value%22:%22vmware%22," \
    + "%22op%22:%22=%22%7D%5D"

uri_decoded = urllib.parse.unquote(uri_original)

print(uri_decoded)
```

The request prompts a response that is structured as shown:

```
https://10.0.0.100/api/hypervisor?from=hlo&pageSize=100&
   sort=[{"property":"name","direction":"ASC"}]&
   filter=[{"property":"type","value":"vmware","op":"="}]
```

Developing a client application with the JupyterLab application

You can develop an API by using JupyterLab, a notetaking application for scripting languages that runs in a web browser. When you develop APIs, you typically work in a command-line shell interface at the beginning, and then switch between the shell session and the editor. By using JupyterLab, you can view all information that you require in a single window to streamline the development process.

Installing, starting, and using JupyterLab

Before you can use JupyterLab for API development, you must install and start the application.

To install JupyterLab, follow the instructions in the online product documentation.

To start JupyterLab, issue the following command:

```
$ jupyter lab &
```

The output is similar to the following example:

```
JupyterLab server extension not enabled, manually loading...
...
[C 15:12:05.230 LabApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=92bd5725f58178bd226ae4303fa0dff0e4ee820815831669
```

Open the URL from a web browser on the workstation where JupyterLab is installed. Verify that you can view the JupyterLab web interface.

To start using JupyterLab, create a notebook with a Python 3 kernel. You can add more kernels such as PHP Hypertext Processor. In the notebook, you can write scripts, run the scripts by using the kernel, and view the result. You can also add texts in a Markdown format.

Using Python Virtual Environments for Jupyter Kernel

If you plan to use non-standard libraries such as Requests and Paramiko, set up a Python Virtual Environment where you can conveniently maintain those libraries. The starting point is to create a Python Virtual Environment. Then, in JupyterLab, you create a kernel that uses this virtual environment.

As an example, assume that you created a Python Virtual Environment and logged in to it:

```
$ python3 -m virtualenv py36_venv
$ source ./py36_venv/bin/active
```

Continuing with this example, you can now add a kernel with the Python Virtual Environment to your JupyterLab application:

```
(py36_venv) $ ipython kernel install --user --name=py36_venv
```

To verify the configuration, you would return to your web browser and ensure that the kernel py36_venv is available in the kernel list. To see a list of kernels with paths, issue the following command:

```
$ jupyter kernelspec list
```

The output is similar to the following example:

```
jupyter kernelspec list
Available kernels:
  py36_venv      /home/sarah/.local/share/jupyter/kernels/py36_venv
```

```
  php             /home/sarah/.ipython/kernels/php
  ruby            /home/sarah/.ipython/kernels/ruby
```

Collaborating with multiple users in a JupyterLab workspace

You can run JupyterLab on a remote server and allow multiple users to access the same workspace. This approach is useful for sharing resources such as JupyterLab notebooks, Python scripts, and virtual environment.

Assume that you want to run JupyterLab on a CentOS Linux server: sarah@10.0.0.110. From your client workstation, establish a Secure Shell (SSH) connection to the server and let localhost use the port 8080 (HTTPS):

```
$ ssh -N -L 8080:localhost:8080 sarah@10.0.0.110
```

Then, establish another SSH connection to this server and run JupyterLab with the following commands:

```
$ ssh sarah@10.0.0.110
```

```
$ jupyter lab --no-browser --port 8080 &
```

From a web browser, open JupyterLab. Ensure that the host name `localhost` on port 8080 connects to the remote server.

Invite other users to connect to the remote server and share your resources with them.

PowerShell REST API

By using the PowerShell scripting language, you can develop a client application that interacts with the IBM Spectrum Protect Plus REST API.

Applicable PowerShell releases: The provided examples are based on PowerShell Core 7.0 and 7.1.

Getting a session ID by using PowerShell

Assume that you want to work with an IBM Spectrum Protect Plus virtual appliance with the following configuration:

- **IPv4 address:** 10.0.0.100
- **Username:** Sarah
- **Password:** `MyPassw0rd!`
- **SSL verification:** False, ignore warnings.

The following PowerShell script loads values to access the REST API.

```
$sppUrl      = "https://10.0.0.100"
$sppUsername = "Sarah"
$sppPassword = ConvertTo-SecureString -String "MyPassw0rd!" -AsPlainText -Force
$sppCred     = New-Object `
   -TypeName System.Management.Automation.PSCredential `
   -ArgumentList ($sppUsername, $sppPassword)

$sppResponse = Invoke-RestMethod `
    -Method Post `
    -Uri $($sppUrl + '/api/endeavour/session') `
    -Authentication Basic `
    -Credential $sppCred `
    -Headers @{
        'Accept'       = 'application/json'
        'Content-type' = 'application/json'
    } `
    -SkipCertificateCheck
```

```
Write-Output $sppResponse
```

The request prompts a response that is structured as shown:

```
sessionid         : db13bc848462424d919afef01ec691cd
user              : @{links=; name=sppadmin; type=NATIVE_USER; ..., id=1000}
demo              : False
preferences       : @{inactivityTimeout=30; ...; minPasswordLength=8}
tenantAdmin       : True
userGroups        : {}
timeZone          : UTC
useServerTime     : False
passwordExpired   : False
usernameExpired   : False
accountDisabled   : False
passwordAge       : 0
passwordExpiresAt : 0
```

After the previous script runs, the following PowerShell snippet is used to retrieve a session ID from the target IBM Spectrum Protect Plus application:

```
$sppSessionid = $sppResponse.sessionid

Write-Output $sppSessionid
```

```
 db13bc848462424d919afef01ec691cd
```

You can use the obtained session ID for subsequent operations. For more information about session IDs, follow the instructions in “Creating a session ID” on page 15.

Sending a request to the REST API by using PowerShell

Assume that you append the following PowerShell snippet to the previous script. The snippet sends an HTTP request to the REST API to display a list of jobs:

```
$sppHeaders = @{
    'X-Endeavour-Sessionid' = $sppSessionid
    'Accept'                = 'application/json'
    'Content-type'          = 'application/json'
    }

$sppResponse = Invoke-RestMethod `
    -Method  Get `
    -Uri     $($sppUrl + '/api/endeavour/job/') `
    -Headers $sppHeaders `
    -SkipCertificateCheck

Write-Output $sppResponse
```

The request prompts a response that is structured as shown:

```
links                          total page jobs
-----                          ----- ---- ----
@{self=; up=; create=; stats=}     4    1 {@{links=; name=Maintenance; ...
```

The jobs object is formatted in JSON. Get this object and convert it to a system array:

```
$sppJobs = $sppResponse.jobs | ConvertTo-JSON | ConvertFrom-JSON

Write-Output $sppJobs
```

The request prompts a response that is structured as shown:

```
links                       : @{self=@{rel=self; href=https://10.0.0.100/...
name                        : Maintenance
description                 : Auto-generated job for maintenance
policyId                    : 1001
...

links                       : ...
name                        : Storage Server Inventory
description                 : Auto-generated job for maintenance
policyId                    : 1002
...

links                       : ...
name                        : Hypervisor Inventory
description                 :
policyId                    : 1003
...

links                       : ...
name                        : Application Server Inventory
description                 :
policyId                    : 1004
...
```

You can retrieve the properties that you require. For example, retrieve the properties `name` and `policyId`:

```
Write-Output $sppJobs | Select-Object -Property name,policyId | Format-Table
```

The request prompts a response that is structured as shown:

```
name                        policyId
----                        --------
Maintenance                 1001
Storage Server Inventory    1002
Hypervisor Inventory        1003
Application Server Inventory 1004
```

The IBM Spectrum Protect Plus REST API has a consistent structure. You can use PowerShell scripts that are similar to this example to develop applications that send requests to the REST API.

Decoding URIs with parameters by using PowerShell

You can decode URI strings with parameters by using a PowerShell snippet.

Assume that you captured the following HTTP request by using Mozilla Firefox:

```
curl 'https://10.0.0.100/api/hypervisor?from=hlo&pageSize=100&
  sort=%5B%7B%22property%22:%22name%22,%22direction%22:%22ASC%22%7D%5D&
  filter=%5B%7B%22property%22:%22type%22,%22value%22:%22vmware%22,
  %22op%22:%22=%22%7D%5D'
  -H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) ...'
...
```

A PowerShell snippet that is similar to the following example can be used to decode the URI string:

```
$uriOriginal = "https://10.0.0.100/api/hypervisor?from=hlo&pageSize=100&" `
  + "sort=%5B%7B%22property%22:%22name%22,%22direction%22:%22ASC%22%7D%5D&" `
  + "filter=%5B%7B%22property%22:%22type%22,%22value%22:%22vmware%22," `
  + "%22op%22:%22=%22%7D%5D"

$uriDecoded = [System.Web.HttpUtility]::UrlDecode($uriOriginal)
Write-Output $uriDecoded
```

The request prompts a response that is structured as shown:

```
https://10.0.0.100/api/hypervisor?from=hlo&pageSize=100&
   sort=[{"property":"name","direction":"ASC"}]&
   filter=[{"property":"type","value":"vmware","op":"="}]
```

Tip: If you receive the error message, "Unable to find type [System.Web.Utility]", ensure that you are using PowerShell Core 7.0 or 7.1. Load the Microsoft .NET Framework class System.Web into the PowerShell session:

```
Add-Type -AssemblyName System.Web
```

Chapter 2. Dashboard and session management

With the REST API, you can complete operations to log in to the IBM Spectrum Protect Plus application and use some widgets in the Dashboard page.

Sessions

You must have a valid session ID for operations such as starting a backup job.

For operations, you also must ensure that users have appropriate permission settings. Permission settings for user accounts are managed by role-based access control. For instruction about managing user accounts, follow the instructions in Chapter 7, “Accounts,” on page 191.

Creating a session ID

You can use the REST API to create a session and obtain the session ID.

Method and URI

To create a session, use a POST method and a URI:

```
POST    https://{hostname|IPv4}/api/endeavour/session
```

This is the only request type of the REST API that does not require a session ID, but does require a username and a password.

Parameters

None.

Example: Create a new session

Assume that you want to create a session for the following IBM Spectrum Protect Plus application and the following user:

- IBM Spectrum Protect Plus
  - **IPv4 address:** 10.0.0.100
  - **SSL authentication:** Disable
- User
  - **User name:** Sarah
  - **Password:** `MyPassw0rd!`

A Python snippet that is similar to the following example can be used to request a new session ID from IBM Spectrum Protect Plus. As you can see, the `headers` section does not include the `X-Endeavour-Sessionid` key-value pair. The script results in a lengthy JSON object with the new session ID.

```
import json
import requests

spp_ipv4 = '10.0.0.100'
spp_username = 'Sarah'
spp_password = 'MyPassw0rd!'
spp_verify = False    # Disable SSL.

# Ignore warning for SSL not being used
from requests.packages.urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)
```

```
_requests = requests.post('https://' + spp_ipv4 + '/api/endeavour/session',
    auth=(spp_username, spp_password),
    headers={
        'Accept':               'application/json',
        'Content-type':         'application/json'
    },
    params="", data="", verify=spp_verify)

_response = json.loads(_requests.text)    # Convert to JSON
```

You will get the session information associated with this user.


```
{
    "sessionid": "43b3b1b2cc434f4eb1b9df8c99e7866f",
    "user": {
        "links": {...},
        "name": "Sarah",
        "type": "NATIVE_USER",
        "typeDisplayName": "Native User",
        "tenantId": 1000,
        "loginCount": 24,
        "lastLogin": 1573480672249,
        "failedLogin": 0,
        "lastFailedLogin": 1572752667593,
        "lastPasswordUpdate": 1564686262606,
        "passwordAge": 0,
        "passwordExpiresAt": 0,
        "passwordExpired": false,
        "accountDisabled": false,
        "accountLocked": false,
        "rbacPath": "user:0/user:1000",
        "permissions": [
            {
                "resourcePool": {
                    "links": {...},
                    "name": "All Resources",
                    "description": "Includes every resource in the system. ",
                    "total": 0,
                    "resources": [
                        {
                            "metadata": {},
                            "include": true,
                            "path": "root:0",
                            "displayName": "Unknown"
                        }
                    ],
                    "rbacPath": "resourcepool:0/resourcepool:1001",
                    "id": "1001"
                },
                "roles": [
                    {
                        "links": {...},
                        "name": "SUPERUSER",
                        "type": "BUILTIN",
                        "description": null,
                        "displayName": "SUPERUSER",
                        "rbacPath": "role:0/role:1000",
                        "id": "1000",
                        "virtualresources": []
                    }
                ],
                "permissionIds": [],
                "roleIds": []
            }
        ],
        "personas": [],
        "metadata": {
            "quickStartAtLogin": true,
            "jobStatusPeriod": 12,
            "whatsNewBanner": "10.1.5",
            "jobLogTypes": [
                "INFO",
                "WARN",
                "ERROR",
                "SUMMARY"
            ],
            "jobLogTypes_JobsOps": [
                "INFO",
                "WARN",
```

```
                    "ERROR",
                    "SUMMARY"
                ],
                "jobHistoryStatusPeriod": 12,
                "initialLogin": 1567668700334,
                "feedbackLastShown": 1570438820557
            },
            "id": "1000"
        },
        "demo": false,
        "tenantAdmin": true,
        "userGroups": [],
        "timeZone": "UTC",
        "useServerTime": false,
        "passwordExpired": false,
        "usernameExpired": false,
        "accountDisabled": false,
        "passwordAge": 0,
        "passwordExpiresAt": 0
    }
```


Normally, you want to get only the session ID, which is represented by the value of `sessionid`:

```
_session_id = _response['sessionid']
print(_session_id)
```

```
43b3b1b2cc434f4eb1b9df8c99e7866f
```

Use this session ID value in the header of future requests:

```
_header={
    'X-Endeavour-Sessionid': _session_id,
    'Accept':                'application/json',
    'Content-type':          'application/json'
}
```

Alerts

Alerts are warning and error messages in the IBM Spectrum Protect Plus environment.

For detailed documentation about messages, see the IBM Spectrum Protect Plus User's and Installation Guide.

In the web user interface, you can see the latest 99 alerts. In the REST API, you can view all alerts.

Getting alert information

You can get a list of alerts and information about each alert.

Tip: To get alert information in CSV format, follow the instructions in “Getting alert information in CSV format” on page 18.

Method and URI

To get information about all alerts in JSON format, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/alert/message
```

Parameters

You can use the operation parameters that are described in “Operation parameters for JSON object output” on page 243.

Data

None.

Example: Get alert information in JSON format

The following Python snippet requests a list of all alerts and information about each alert.

```
requests.get('https://' + spp_ipv4 + '/api/endeavour/alert/message',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "total": 2584,
    "unacknowledged": 467,
    "page": 1,
    "alerts": [
        {
            "links": {...},
            "name": "JOB_PARTIAL",
            "alertTime": 1564688662492,
            "category": "JOB",
            "categoryDisplayName": "JOB",
            "type": "WARN",
            "typeDisplayName": "WARN",
            "initialMessage": "Job Application Server Inventory (id=1003, sessio
   n=1,564,688,656,142) partially succeeded.",
            "message": "Job Application Server Inventory (id=1003, session=1,564
   ,688,656,142) partially succeeded.",
            "messageName": "ALERT_JOB_PARTIALLY_SUCCEEDED",
            "messageParams": [
                "Application Server Inventory",
                "1003",
                1564688656142
            ],
            "initMessageParams": [
                "Application Server Inventory",
                "1003",
                1564688656142
            ],
            "dataSource": "Application Server Inventory:1564688656142",
            "status": "ACTIVE",
            "statusDisplayName": "ACTIVE",
            "storageId": null,
            "serverId": null,
            "jobId": "1003",
            "jobSessionId": 1564688656142,
            "retention": 180,
            "first": 1564688662492,
            "last": 1564688662492,
            "expiresAt": 1580240662519,
            "count": 1,
            "acknowledged": true,
            "expired": false,
            "unique": true,
            "id": "5d4341166dde3d12afd8b11f"
        },
        {...}, {...}, ..., {...}
    ]
}
```

Getting alert information in CSV format

You can get a list of alerts in CSV format. The CSV data will be shown in the response body.

Tip: This feature is not available in the web user interface.

Method and URI

To get information about all alerts, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/alert/message/download/csv
```

Parameters

None.

Data

None.

Example: Get alert information in CSV format

In this example, a Python snippet is used to get information about alerts in CSV format.

```
_response = requests.get('https://' + spp_ipv4
    + '/api/endeavour/alert/message/download/csv',
    headers={...}, verify=...)

print(_response.text)
```

The request prompts a response as shown, with the HTTP status of 200 (OK). The columns `Time` and `Last` are in the UNIX time format followed by three digits, which represent milliseconds.

Tip: The response body can be extremely large.

```
Time,ReadableTime,Name,Category,Type,DataSource,First,ReadableFirst,Last,Readabl
   eLast,Level,Message
1564775266973,Fri 2 Aug 2019 19:47:46 UTC,JOB_FAILED,JOB,ERROR,onDemandRestore_1
   564775110936:1564775111170,1564775266973,Fri 2 Aug 2019 19:47:46 UTC,1564775266
   973,Fri 2 Aug 2019 19:47:46 UTC,0,Job onDemandRestore_1564775110936 (id=1007 se
   ssion=1564775111170) failed.
...
```

You can load a CSV table into spreadsheet applications, such as Microsoft Excel. To learn how to load a CSV table into a spreadsheet, see the documentation for the spreadsheet application.

Chapter 3. Jobs and Operations

Most actions in IBM Spectrum Protect Plus are defined as jobs.

For example, assume that you associate some VMware instances with an SLA Gold policy. IBM Spectrum Protect Plus creates a job, names it "vmware_Gold", and starts this job at the scheduled day and time, as defined in the SLA policy. You can pause and resume this scheduled job, and you can start it manually as well. The job can run multiple times, and for each time, IBM Spectrum Protect Plus creates and manages a job session. Every job session has a job log that contains records of actions that were taken. If a job session fails, you can see the job log and identify the cause of the failure.

Running Jobs

You can get a list of active and ended jobs with brief information for each job. You can also cancel a running job.

Getting job session information

You can get information about job sessions. IBM Spectrum Protet Plus assigns a `{jobsessionId}` for every job and stores the information about the job. The information includes the type of job, the start time, the end time, and the job status: `RUNNING`, `PARTIAL`, `COMPLETED`, and so on.

Method and URI

To get information about job sessions, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/jobsession
```

To get information about a specific job session, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/jobsession/{jobsessionId}
```

You can get the `{jobsessionId}` of the specific job session from the list of all job sessions.

Parameters

Parameter 1: `pageStartIndex`

By default, the REST API splits the response body every 100 records. To see the next 100 records, use this parameter to see the other 100 records from the given index value. Index values are assigned to all records, starting with zero. Therefore, to see the 101st through the 200th records, use `100` for the pageStartIndex value.

- **Value Example:** `100`
- **Type:** String. Available in the web user interface.

Parameter 2: `filter`

Filter the results by specifying criteria.

- **Example value:** Use the following value to get results that has the all status types:

```
[
     {
         "property": "status",
         "value":    [
             "RUNNING", "PENDING", "HELD", "WAITING", "PAUSING", "PAUSED",
             "STOPPING", "STOPPED", "CANCELING", "CANCELED", "SUSPENDING",
             "SUSPENDED", "FAILED", "PARTIAL", "COMPLETED", "UNKNOWN"
```

```
            ],
            "op":       "IN"
        }
    ]
```

- **Type:** List. Available in the web user interface.

Parameter 3: `sort`

By default, job session records are sorted by the start time, in ascending order, from the oldest record to the latest record. Use this parameter to sort the job session records in descending order.

- **Value:**

```
    [
        {
            "property":  "start",
            "direction": "DESC"
        }
    ]
```

- **Type:** Array. Available in the web user interface.

Data

None.

Example 1: Get information about all job sessions

Assume that you want to get a list of all job session records. Also assume that you want to sort job session records by the start time, in descending order, so that the latest jobs are at the top of the results.

The following Python snippet requests a list of all job session records and sorts them by start time in descending order:

```
_params = (
    ('sort', '[\
        {\
            "property": "start",\
            "direction":"DESC"\
        }\
    ]'),
)

requests.get('https://' + spp_ipv4 + '/api/endeavour/jobsession',
    headers={...}, params=_params, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "total": 4877,
    "page": 1,
    "sessions": [
        {
            "links": {...},
            "jobName": "Application Server Inventory",
            "jobId": "1003",
            "type": "catalog",
            "typeDisplayName": "Catalog",
            "subType": "application",
            "subTypeDisplayName": "Application",
            "subPolicyType": null,
            "subPolicyTypeDisplayName": null,
            "serviceId": "serviceprovider.catalog.application",
            "displayName": null,
            "start": 1577881234,
            "end": 1577882234123,
            "duration": 1000000,
            "status": "PARTIAL",
            "statusDisplayName": "Partial",
```

```
            "indexStatus": "COMPLETED",
            "description": "Autorun - 10.4.4.1 only",
            "results": "Started",
            "properties": null,
            "numTasks": 2,
            "previousLastTask": 0,
            "lastUpdate": 1577882235123,
            "percent": 0,
            "expiresAt": null,
            "expirationTime": null,
            "retention": 0,
            "retentionUnit": null,
            "expired": false,
            "hasCatalog": false,
            "hasCondensed": false,
            "wormProtected": false,
            "rerunnable": false,
            "userUpdate": false,
            "actions": [],
            "statistics": null,
            "policySnapshot": {
                "id": "1003",
                "name": "Application Server Inventory",
                "type": "catalog",
                "subType": "application",
                "serviceId": "serviceprovider.catalog.application",
                "version": null,
                "spec": {
                    "source": [
                        {
                            "href": "http://localhost:8082/api/appserver/1001",
                            "resourceType": "appserver",
                            "name": "10.4.4.1",
                            "id": "1001",
                            "include": true,
                            "metadata": {
                                "path": ":",
                                "name": "10.4.4.1"
                            }
                        }
                    ],
                    "option": {
                        "retention": "3",
                        "maxtasks": 50
                    },
                    "notification": [],
                    "applicationtype": "oracle"
                },
                "description": "",
                "script": null
            },
            "id": "1564688656142"
        },
        {...}, ..., {...}
    ]
}
```


The first job session record in the response body contains information that includes the following items:

- **Job name:** Application Server Inventory
- **Start time:** 1577881234123 (1 January 2020 at 6:20:34:123 AM US-CST)
- **End time:** 1577882234123 (1 January 2020 at 6:37:14:123 AM US-CST)
- **Duration:** 1,000,000 milliseconds (16 minutes and 40 seconds)
- **Status:** Partial
- `{jobsessionId}`: 1564688656142

The `total` value in the response indicates that there are 4,877 job session records. By default, the REST API splits the response body every 100 records. The response contains only the first 100 job session records. To see the next 100 records, use the following Python snippet:

```
_params = {
    "pageStartIndex": 100
}
```

```
requests.get('https://' + spp_ipv4 + '/api/endeavour/jobsession',
    headers={...}, params=_params, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200. As you see in the response body below, the page number increased to 2.

```
{
    "links": {...},
    "total": 4877,
    "page": 2,
    "sessions": [{...}, ..., {...}]
}
```

To see page 3 for the records of the 201st through the 300th, increase the `pageStartIndex` value to 200.

Example 2: Get information about a specific job session

Assume that you want to track one of the job sessions, and you have the `{jobsessionId}`: 1564688656142.

A Python snippet that is similar to the following example can be used to request the information about this job session:

```
jobsession_id = "1564688656142"

requests.get('https://' + spp_ipv4 + '/api/endeavour/jobsession/'
    + jobsession_id,
    headers={...}, verify=...)
```

You will get the job session record, with the HTTP status of 200 (OK).

Example 3: Get information about job sessions that are running

Assume that you want to get information about job sessions that are running or that will run shortly. In this case, you have to use the `filter` parameter to select only the job sessions that have the status value of `RUNNING`, `PENDING`, `HELD`, or `CANCELING`.

A Python snippet that is similar to the following example can be used to provide the filtered information:

```
_params = (
    ('sort',   '[\
        {\
           "property":  "start",\
           "direction": "DESC"\
        }\
    ]'),
    ('filter', '[\
        {\
            "property": "status",\
            "value":    ["RUNNING", "PENDING", "HELD", "CANCELING"],\
            "op":       "IN"\
        }\
    ]'),
)

requests.get('https://' + spp_ipv4 + '/api/endeavour/jobsession',
    headers={...}, params=_params, verify=...)
```

The request prompts a response that is similar to the response body in Example 1, with the HTTP status of 200 (OK). All of the job session records have the status value of `RUNING`, `PENDING`, `HELD`, or `CANCELING`.

Canceling a job

You can cancel a job that has the status of running, pending, or paused schedule. If you cancel a running backup job, you can retain the partial data backup.

Method and URI

To cancel a job, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/api/endeavour/jobsession/{jobsessionId}
```

Tip: To get a `{jobsessionId}`, follow the instructions in "Getting job session information" on page 21.

Parameters

Parameter: `action`

You must specify the action parameter to cancel the job.

- **Value:** `cancel`
- **Type:** String. Required. Available in the web user interface.

Data

Data: `catalogCompletedObjects`

For backup jobs, enable this value to keep the data that was backed up.

- **Example value:** true
- **Type:** Boolean. Available in the web user interface.

Example: Cancel a job

A Python snippet that is similar to the following example can be used to send a request to IBM Spectrum Protect Plus to cancel the backup job (`{jobsession_id}` 1577841955375).

```
jobsession_id = "1577841955375"

_params = {"action": "cancel"}

_data = f'''{{
    "catalogCompletedObjects": true
}}'''

requests.post('https://' + spp_ipv4 + '/api/endeavour/jobsession/'
    + jobsession_id,
    headers={...}, params=_params, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 201 (Created). The job status must change from running to canceling.

Job History

Each job is done by proceeding a series of actions. These actions are recorded in a job log which is created for each job. You can troubleshoot unsuccessful jobs by seeing their job logs and identify root causes.

Getting a log for a job session

You can get a detailed log for a job session. You can get a log in JSON format or CSV format.

Method and URI

To get a log of job sessions in JSON format, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/log/job
```

To get a log of job sessions in CSV format, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/log/job/download/csv
```

Tip: To get a `{jobsessionId}`, follow the instructions in "Getting job session information" on page 21.

Parameters

Parameter: `filter`

Filter results by specifying criteria.

- **Example value:** Use the following value to get results for a job session with a `{jobsessionId}` of 1577841955375:

```
"filter": [
    {
        "property": "jobsessionId",
        "value":    "1577841955375",
        "op":       "="
    }
]
```

- **Type:** JSON object. Required.

Data

None.

Example: Get a log for a job session in CSV format

A Python snippet that is similar to the following example can be used to request a log for a job session (`{jobsessionId}` 1577841955375):

```
jobsession_id = "1577841955375"

_params = {
    "filter": f'''[
        {{
            "property": "jobsessionId",
            "value":    "{jobsession_id}",
            "op":       "="
        }}
    ]'''
}

_response = requests.get('https://' + spp_ipv4
    + '/api/endeavour/log/job/download/csv',
    headers={...}, params=_params, verify=...)
```

```
print(_response.text)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
Type,Time,Server Time,Task ID,Message ID,Message
SUMMARY,1577841955375,Wed 1 Jan 2020 01:25:55 UTC,,CTGGA2399,Starting job ...
DETAIL,1577841957287,Wed 1 Jan 2020 01:25:57 UTC,2,CTGGA0171,Job options : ...
...
```

Getting progress information about job sessions

You can get progress information about job sessions.

Method and URI

To get a list of jobs with progress information for each item, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/progress
```

Parameters

Parameter 1: `pageSize`

Specify the maximum number of objects for each page.

- **Example value:** 100
- **Type:** Integer.

Parameter 2: `filter`

Filter results by specifying criteria.

- **Example value:** Assume that you want to filter jobs by using the following criteria:
  - **Job status:** Failed, aborted, partial, or canceled
  - **Job types:** Virtualized systems
  - `{jobsessionId}`: 1610664896075

You can use the following JSON object:

```
"filter": [
    {
        "property": "summaryStatus",
        "value":    "[\"complete\",\"partial\",\"failed\",\"skipped\"]",
        "op":       "IN"
    },
    {
        "property": "type",
        "value":    "HYPERVISORVM",
        "op":       "="
    },
    {
        "property": "sessionId",
        "value":    "1610664896075",
        "op":       "="
    }
]
```

- **Type:** List.

For information about operations parameters for JSON object output, filtering, sorting, and pagination, follow the instructions in “Operation parameters for JSON object output” on page 243.

Parameter 2.1: `filter` > List > JSON object: `summaryStatus`

Filter previous jobs based on status. You can use the following values for the `summaryStatus` key:

summary Status value	Job status
complete	Completed
partial	Partial
skipped	Canceled
failed	Failed or aborted
running	Running
pending	Pending

- **Type:** JSON object.

Parameter 2.2: `filter` > List > JSON object: `type`

Filter jobs based on type. Use the following values for the `type` key:

type value	Job type
HYPERVISORVM	Virtualized systems (VMware vCenter, Hyper-V, and Amazon Elastic Compute Cloud, EC2)
INSTANCES	Cloud management
DATABASE	(All other types)

- **Type:** JSON object.

Parameter 2.3: `filter` > List > JSON object: `sessionId`

Filter JSON objects based on `{jobsessionId}`.

- **Type:** JSON object.

Tip: To get a `{jobsessionId}`, follow the instructions in "Getting job session information" on page 21.

Parameter 3: `sort` > List > JSON object: `id`

Sort job items by IDs, which are based on the start time of the job.

- **Example value:** Use the following JSON object to sort the job items by their start time in descending order:

```
{
    "property": "id",
    "value":    "DESC"
}
```

- **Type:** JSON object. Available in the web user interface.

Data

None.

Example 1: Get progress information about incomplete jobs for virtualized systems

Assume that IBM Spectrum Protect Plus was used to run jobs, such as backup jobs for VMware vCenter objects, and you want to get progress information based on the following criteria:

- **Pagination:** 10 records per page, get the records between the 1st and 10th

- **Job status:** Failed, aborted, partial, or canceled
- **Job types:** Virtualized systems
- **Sorting order:** Sort the records by their start time in ascending order

Tip: The response body can be extremely large. Use filters and pagination.

A Python snippet that is similar to the following example can be used to request a list of jobs with progress information:

```
_params = {
    "pageSize": 10,
    "pageStartIndex": 0,
    "filter": {
        json.dumps([
            {
                "property": "summaryStatus",
                "value":    "[\"failed\", \"partial\", \"skipped\"]",
                "op":       "IN"
            },
            {
                "property": "type",
                "value":    "HYPERVISORVM",
                "op":       "="
            }
        ])
    },
    "sort": {
        json.dumps([
            {
                "property":  "id",
                "direction": "ASC"
            }
        ])
    }
}

_response = requests.get('https://' + spp_ipv4
    + '/api/progress',
    headers={...}, params=_params, verify=...)

print(_response.text)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {
        "self": {
            "rel": "self",
            "href": "https://10.0.0.100/api/progress",
            "hreflang": null,
            "media": null,
            "title": null,
            "type": null,
            "deprecation": null
        }
    },
    "page": 1,
    "total": 20,
    "summaries": [
        {
            "links": {
                "up": {
                    "rel": "self",
                    "href": "https://10.0.0.100/api/progress",
                    "hreflang": null,
                    "media": null,
                    "title": null,
                    "type": null,
                    "deprecation": null
                }
            },
            "name": "Service-Dallas",
            "type": "HYPERVISORVM",
            "association": null,
            "protectionInfo": null,
```

```
            "recoveryInfo": null,
            "recoveryPoint": null,
            "siteId": "10.0.1.1",
            "autoProvisionedsubPolicyName": null,
            "rbacPath": null,
            "rbacPathId": null,
            "extInfo": null,
            "location": "/vmware-dallas1.bluemachines.com.invalid/Production/Ser
        vice-Dallas",
            "metadataPath": null,
            "inHLO": true,
            "jobPK": "1096_1610664896075",
            "resourceName": "Service-Dallas",
            "resourcePk": "221199d6aa263807e7387a0bfcb15326",
            "proxyUsed": "10.0.1.1",
            "proxyUsedId": "2102",
            "isBase": true,
            "vmdkIds": null,
            "vmdkName": null,
            "transportMethod": "[hotadd]",
            "vmdkCount": 1,
            "protectedVmdkCount": 1,
            "totalDataTransferred": "37.59 GB",
            "total": 0,
            "throughput": "217081904",
            "queueTimeStart": 0,
            "queueTimeEnd": 0,
            "isSuccess": true,
            "summaryStatus": "partial",
            "vSnapUsed": "10.0.1.1",
            "vSnapUsedId": null,
            "percentageCompleted": 50,
            "vsnapVolumes": {
                "spp_1096_2207_176f26b6caf__group66c64994fb89402d922491630feba43
        ": "2102.volume.3"
            },
            "groupParentPk": null,
            "groupPercentageCompleted": 0,
            "resourcePhaseName": null,
            "resourcePhaseProgressPercentage": 0,
            "phaseNumber": 0,
            "totalPhases": 0,
            "Name": "Service-Dallas",
            "partitionId": null,
            "message": null,
            "backupStartTime": 1610384909,
            "backupEndTime": 1610385140,
            "errorCode": 0,
            "elapsedTime": 230,
            "base": true,
            "success": true,
            "protected": false,
            "failed": false,
            "systemHold": false,
            "id": "1610664896075.221199d6aa263807e7387a0bfcb15326",
            "host": "10.250.250.13",
            "resourceType": "resourcesummary",
            "resourceSummary": {
                "name": "Service-Dallas",
                "type": "HYPERVISORVM",
                "association": null,
                "protectionInfo": null,
                "recoveryInfo": null,
                "recoveryPoint": null,
                "siteId": "10.0.1.1",
                "autoProvisionedsubPolicyName": null,
                "rbacPath": null,
                "rbacPathId": null,
                "extInfo": null,
                "location": "/vmware-dallas1.bluemachines.com.invalid/Production
        /Service-Dallas",
                "metadataPath": null,
                "inHLO": true,
                "jobPK": "1096_1610664896075",
                "resourceName": "Service-Dallas",
                "resourcePk": "221199d6aa263807e7387a0bfcb15326",
                "proxyUsed": "10.0.1.1",
                "proxyUsedId": "2102",
                "isBase": true,
                "vmdkIds": null,
                "vmdkName": null,
                "transportMethod": "[hotadd]",
```

```
                    "vmdkCount": 1,
                    "protectedVmdkCount": 1,
                    "totalDataTransferred": "37.59 GB",
                    "total": 0,
                    "throughput": "217081904",
                    "queueTimeStart": 0,
                    "queueTimeEnd": 0,
                    "isSuccess": true,
                    "summaryStatus": "complete",
                    "vSnapUsed": "10.0.1.1",
                    "vSnapUsedId": null,
                    "percentageCompleted": 50,
                    "vsnapVolumes": {
                        "spp_1096_2207_176f26b6caf__group66c64994fb89402d922491630fe
      ba43": "2102.volume.3"
                    },
                    "groupParentPk": null,
                    "groupPercentageCompleted": 0,
                    "resourcePhaseName": null,
                    "resourcePhaseProgressPercentage": 0,
                    "phaseNumber": 0,
                    "totalPhases": 0,
                    "Name": "Service-Dallas",
                    "partitionId": null,
                    "message": null,
                    "backupStartTime": 1610384909,
                    "backupEndTime": 1610385140,
                    "errorCode": 0,
                    "elapsedTime": 230,
                    "base": true,
                    "success": true,
                    "protected": false,
                    "failed": false,
                    "systemHold": false,
                    "id": "1610664896075.221199d6aa263807e7387a0bfcb15326",
                    "host": "10.250.250.13"
                }
            },
            {...}, ..., {...}
    ]
}
```

Example 2: Get progress information about a job

Assume that you want to get progress information about a job with {sessionId} 1610664896075.

A Python snippet that is similar to the following example can be used to request progress information:

```
_params = {
    "filter": {
        json.dumps([
            {
                "property": "sessionId",
                "value":    "1610664896075",
                "op":       "="
            }
        ])
    }
}

_response = requests.get('https://' + spp_ipv4
    + '/api/progress',
    headers={...}, params=_params, verify=...)
```

The request prompts a response that is similar to the response body in the previous example.

Schedule

You can schedule jobs, or you can start jobs manually.

Getting information about scheduled jobs

You can get information about scheduled jobs.

Method and URI

To get information about all scheduled jobs, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/job
```

To get information about a specific scheduled job, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/job/{jobId}
```

Tip: To get a {jobId} value based on the name of the scheduled job, follow the instructions in “Getting a {jobId}” on page 234.

Parameters

None.

Data

None.

Example 1: Get information about all scheduled jobs

A Python snippet that is similar to the following example can be used to request a list of all scheduled jobs and their information.

```
requests.get('https://' + spp_ipv4 + '/api/endeavour/job',
    headers={...}, verify=...)
```


```
{
    "links": {...},
    "total": 18,
    "page": 1,
    "jobs": [
        {
            "links": {...},
            "name": "Maintenance",
            "description": "Auto-generated job for maintenance",
            "policyId": "1001",
            "policyName": "Maintenance",
            "type": "maintenance",
            "typeDisplayName": "Maintenance",
            "subType": "maintenance",
            "subTypeDisplayName": "Maintenance",
            "serviceId": "com.syncsort.dp.xsb.serviceprovider.maintenance",
            "displayName": "Maintenance",
            "status": "IDLE",
            "statusDisplayName": "Idle",
            "lastSessionStatus": "COMPLETED",
            "lastSessionStatusDisplayName": "Completed",
            "triggerIds": [
                "1001"
            ],
            "triggerData": [
                {
                    "triggerId": "1001",
                    "triggerInfo": null
                }
            ],
            "lastRunTime": 1589151600156,
            "nextFireTime": 1589238000000,
            "lastSessionDuration": 5135140,
            "tenantId": 1000,
```

```
            "actions": null,
            "statistics": null,
            "lastrun": {
                "sessionId": "1589151600106",
                "jobName": "Maintenance",
                "type": "maintenance",
                "subType": "maintenance",
                "serviceId": "serviceprovider.maintenance",
                "start": 1589151600156,
                "end": 1589156735296,
                "duration": 5135140,
                "status": "COMPLETED",
                "results": "COMPLETED",
                "properties": null,
                "numTasks": 14,
                "previousLastTask": 0,
                "lastUpdate": 1589156735306,
                "percent": 100,
                "policySnapshot": null
            },
            "id": "1001"
        },
        {
            "links": {...},
            "name": "Storage Server Inventory",
            "description": "Auto-generated job for maintenance",
            "policyId": "1002",
            "policyName": "Storage Server Inventory",
            "type": "catalog",
            "typeDisplayName": "Catalog",
            "subType": "storage",
            "subTypeDisplayName": "Storage",
            "serviceId": "serviceprovider.catalog.storage",
            "displayName": "Storage Server Inventory",
            "status": "IDLE",
            "statusDisplayName": "Idle",
            "lastSessionStatus": "PARTIAL",
            "lastSessionStatusDisplayName": "Partial",
            "triggerIds": [
                "1002"
            ],
            "triggerData": [
                {
                    "triggerId": "1002",
                    "triggerInfo": null
                }
            ],
            "lastRunTime": 1589145300278,
            "nextFireTime": 1589231700000,
            "lastSessionDuration": 26231,
            "tenantId": 1000,
            "actions": null,
            "statistics": null,
            "lastrun": {...},
            "id": "1002"
        },
        {
            "links": {...},
            "name": "Application Server Inventory",
            "description": null,
            "policyId": "1003",
            "policyName": "Application Server Inventory",
            "type": "catalog",
            "typeDisplayName": "Catalog",
            "subType": "application",
            "subTypeDisplayName": "Application",
            "serviceId": "serviceprovider.catalog.application",
            "displayName": "Application Server Inventory",
            ...,
            "id": "1003"
        },
        {
            "links": {...},
            "name": "Hypervisor Inventory",
            "description": null,
            "policyId": "1004",
            "policyName": "Hypervisor Inventory",
            "type": "catalog",
            "typeDisplayName": "Catalog",
            "subType": "hypervisor",
            "subTypeDisplayName": "Hypervisor",
            ...,
```

```
            "id": "1004"
        },
        {
            "links": {...},
            "name": "oracle_Gold",
            "description": "",
            "policyId": "1005",
            "policyName": "oracle_Gold",
            "type": "protection",
            "typeDisplayName": "Protection",
            "subType": "oracle",
            "subTypeDisplayName": "Oracle",
            "serviceId": "serviceprovider.protection.application",
            "displayName": "Application / File System Protection",
            ...
            "id": "1005"
        },
        {
            "links": {...},
            "name": "oracle_Silver",
            ...
            "id": "1006"
        },
        {
            "links": {...},
            "name": "sql_Gold",
            "description": "",
            "policyId": "1011",
            "policyName": "sql_Gold",
            "type": "protection",
            "typeDisplayName": "Protection",
            "subType": "sql",
            "subTypeDisplayName": "SQL",
            "serviceId": "serviceprovider.protection.application",
            "displayName": "Application / File System Protection",
            ...,
            "id": "1011"
        },
        {...},
        {
            "links": {...},
            "name": "hyperv_Gold",
            "description": "",
            "policyId": "1014",
            "policyName": "hyperv_Gold",
            "type": "protection",
            "typeDisplayName": "Protection",
            "subType": "hyperv",
            "subTypeDisplayName": "Hyper-V",
            "serviceId": "serviceprovider.protection.hypervisor",
            "displayName": "Hypervisor Backup",
            ...,
            "id": "1014"
        },
        {
            "links": {...},
            "name": "Report_vSnap Storage Utilization",
            "description": "Auto-generated job for Policy Report_vSnap Storage Utilization",
            "policyId": "1023",
            "policyName": "Report_vSnap Storage Utilization",
            "type": "analyze",
            "typeDisplayName": "Analyze",
            "subType": "report",
            "subTypeDisplayName": "Report",
            "serviceId": "serviceprovider.report",
            "displayName": "Report",
            ...,
            "id": "1023"
        },
        {
            "links": {...},
            "name": "Report_vSnapStorageUtilization-Dallas1",
            "description": "Auto-generated job for Policy Report_vSnapStorageUtilization-
Dallas1",
            "policyId": "1028",
            "policyName": "Report_vSnapStorageUtilization-Dallas1",
            ...,
            "id": "1024"
        },
        ...,
        {
            "links": {...},
```

```
            "name": "vmware_Diamond",
            ...
            "id": "1043"
        },
        ...
    ]
}
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

Example 2: Get information about a specific scheduled job

Assume that you added an scheduled job, Diamond ({slapolicyId} 2101), to IBM Spectrum Protect Plus.

A Python snippet that is similar to the following example can be used to request information about this scheduled job:

```
job_id = "1043"

requests.get('https://' + spp_ipv4 + 'api/endeavour/job/' + job_id,
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "name": "vmware_Diamond",
    "description": "",
    "policyId": "1050",
    "policyName": "vmware_Diamond",
    "type": "protection",
    "typeDisplayName": "Protection",
    "subType": "vmware",
    "subTypeDisplayName": "VMware",
    "serviceId": "serviceprovider.protection.hypervisor",
    "displayName": "Hypervisor Backup",
    "status": "IDLE",
    "statusDisplayName": "Idle",
    "lastSessionStatus": "PARTIAL",
    "lastSessionStatusDisplayName": "Partial",
    "triggerIds": [
        "1049"
    ],
    "triggerData": [
        {
            "triggerId": "1049",
            "triggerInfo": {
                "subpolicy_id": "VM Replication0(2113)"
            }
        }
    ],
    "lastRunTime": 1589176802081,
    "nextFireTime": 1589263200000,
    "lastSessionDuration": 8154,
    "tenantId": 1000,
    "actions": null,
    "statistics": null,
    "lastrun": {
        "sessionId": "1589176800026",
        "jobName": "vmware_Diamond",
        "type": "protection",
        "subType": "vmware",
        "serviceId": "serviceprovider.protection.hypervisor",
        "start": 1589176802081,
        "end": 1589176810235,
        "duration": 8154,
        "status": "PARTIAL",
        "results": null,
        "properties": {
            "statistics": [
                {
                    "resourceType": "vm",
                    "total": 0,
                    "success": 0,
```

```
                        "failed": 0,
                        "skipped": 0,
                        "names": null,
                        "jobSessionId": 0,
                        "policyId": null,
                        "startTime": 0
                    },
                    {
                        "resourceType": "datastore",
                        "total": 0,
                        "success": 0,
                        "failed": 0,
                        "skipped": null,
                        "names": null,
                        "jobSessionId": 0,
                        "policyId": null,
                        "startTime": 0
                    }
                ]
            },
            "numTasks": 2,
            "previousLastTask": 0,
            "lastUpdate": 1589176810260,
            "percent": 0,
            "policySnapshot": null
        },
        "id": "1043"
    }
```


Starting a job manually

IBM Spectrum Protect Plus IBM Spectrum Protect Plus can start backup jobs that are associated with service level agreement (SLA) policies in two ways:

- An SLA policy for associated instances of hypervisors, application servers, and IBM Spectrum Protect Plus catalogs. Backup jobs are started automatically after the SLA policy is defined.
- An SLA policy for associated instances of specific types of virtualized systems, application servers or IBM Spectrum Protect Plus. Backup jobs must be started manually regardless of how the SLA policy is defined.

For example, assume that you have a custom SLA policy, Diamond, which starts every day at midnight UTC. You associate the SLA policy with some instances of VMware, Oracle Database, and IBM Db2.

In this case, you do not have to create a script to start this SLA policy because backup jobs are started automatically by IBM Spectrum Protect Plus. But in some cases, you want to start the backup jobs *manually*. A manual start is supported for some types of virtualized systems, application servers, and IBM Spectrum Protect Plus. In that case, the SLA policy name has one of the following prefixes:

Prefix	**Product**
vmware	VMware vCenter
hyperv	Hyper-V
oracle	Oracle Database
sql	SQL Server
db2	Db2
exch	Exchange Server
mongo	MongoDB
office365	Microsoft 365
catalog	IBM Spectrum Protect Plus

Method and URI

To start a job manually, use a POST method and a URI:

```
POST    https://{hostname|IP}/api/endeavour/job/{slapolicyId}
```

Tip: To get an {slapolicyId} value based on its name, follow the instructions in "Getting an {slapolicyId}" on page 241.

Parameters

Parameter 1: `action`

Specify the type of action.

- **Value:** `inventory`
- **Type:** System string. Required. Available in the web interface.

Parameter 2: `actionname`

Specify the step of the action.

- **Value:** `start`
- **Type:** System string. Required. Available in the web interface.

Data

None.

Example: Start a job manually

Assume that you want to start a job, vmware_Gold (`{slapolicyId}` 1021), manually.

A Python snippet that is similar to the following example can be used to start a job manually:

```
sla_id = "1021"    # vmware_Gold

_params = {
    "action":     "start",
    "actionname": "start"
}

requests.post('https://' + spp_ipv4 + '/api/endeavour/job/' + sla_id,
     headers={...}, params=_params, data="", verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "name": "vmware_Gold",
    "description": "Auto-generated job for Policy vmware_Gold",
    "policyId": "1021",
    "policyName": "vmware_Gold",
    "type": "protection",
    "typeDisplayName": "Protection",
    "subType": "vmware",
    "subTypeDisplayName": "VMware",
    "serviceId": "serviceprovider.protection.hypervisor",
    "displayName": "Hypervisor Backup",
    "status": "RESOURCE ACTIVE",
    "statusDisplayName": "Resource active",
    "lastSessionStatus": "PARTIAL",
    "lastSessionStatusDisplayName": "Partial",
    "triggerIds": [
        "1015"
    ],
    "triggerData": [
        {
```

```
            "triggerId": "1015",
            "triggerInfo": {
                "subpolicy_id": "VADP Backup(2000)"
            }
        }
    ],
    "lastRunTime": 1576153580805,
    "nextFireTime": 1576167979000,
    "lastSessionDuration": 3572,
    "tenantId": 1000,
    "actions": null,
    "statistics": null,
    "policy": {
        "links": {...},
        "name": "vmware_Gold",
        "type": "protection",
        "subType": "vmware",
        "serviceId": "serviceprovider.protection.hypervisor",
        "description": "",
        "version": "3.0",
        "rbacPath": "root:0/policy:0/policytype:serviceprovider.protection.hyper
visor/policy:1021",
        "tenantId": 1000,
        "creationTime": 1570192798811,
        "lastUpdated": 0,
        "spec": {
            "source": [],
            "storageworkflow": [
                {
                    "id": "2000",
                    "name": "Gold",
                    "href": "http://localhost:8082/api/spec/storageprofile/2000"
                }
            ],
            "option": {
                "maxtasks": 0,
                "vmsnapshot": {
                    "takesnapshot": false,
                    "concurrentsnapshotsonesx": 3,
                    "includememory": false,
                    "consistency": false,
                    "consistencyvms": [],
                    "scriptvms": [],
                    "truncateapplicationlogs": false
                },
                "skipreadonlydatastores": true
            },
            "notification": []
        },
        "logicalDelete": false,
        "script": {
            "preScript": null,
            "postScript": null,
            "preSnap": null,
            "postSnap": null,
            "preGuest": null,
            "postGuest": null,
            "preGuestSnap": null,
            "postGuestSnap": null,
            "continueScriptsOnError": false
        },
        "id": "1021"
    },
    "triggers": [
        {
            "links": {...},
            "name": "1021__VADP Backup(2000)",
            "type": "Schedule",
            "category": "INTERNAL",
            "triggerInfo": {
                "id": 1015,
                "name": "1015:1021",
                "comment": null,
                "creatorId": 0,
                "creationTime": 1570192799690,
                "type": null,
                "flags": 0,
                "pk": null,
                "triggerId": "1015",
                "jobId": "1021",
                "triggerInfo": {
                    "subpolicy_id": "VADP Backup(2000)"
```

```
                    },
                    "category": "INTERNAL"
                },
                "properties": {
                    "activateDate": 1564604779173,
                    "deactivateDate": null,
                    "frequency": 4,
                    "type": "HOURLY",
                    "dowList": [
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false
                    ],
                    "domList": [
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false,
                        false
                    ]
                },
                "rbacPath": "trigger:0/trigger:1015",
                "onHold": false,
                "id": "1015"
            }
        ],
        "lastrun": {
            "sessionId": "1576153579173",
            "jobName": "vmware_Gold",
            "type": "protection",
            "subType": "vmware",
            "serviceId": "serviceprovider.protection.hypervisor",
            "start": 1576153580805,
            "end": 1576153584377,
            "duration": 3572,
            "status": "PARTIAL",
            "results": null,
            "properties": {
                "statistics": []
            },
            "numTasks": 2,
            "previousLastTask": 0,
            "lastUpdate": 1576153584402,
            "percent": 5,
            "policySnapshot": null
        },
        "id": "1021"
    }
```

Starting an ad hoc backup job

There are two scenarios to start a backup job for VMware and Microsoft Hyper-V instances:

- Associate an SLA policy with virtualized system instances (for example, VMware virtual machines) so that the backup job can be started automatically or manually.
- Start an ad hoc backup job.

The ad hoc backup does not have to include all instances that are associated with the SLA policies.

A single ad hoc backup job can process multiple virtual machines of the same type of virtualized system: VMware vCenter, Microsoft Hyper-V, or Amazon EC2. You cannot specify folders or datacenters even if they are associated with the specific SLA policy. You cannot specify virtual machines hosted on VMware *and* Microsoft Hyper-V at the same time.

Method and URI

To start an ad hoc backup, use a POST method and a URI:

```
POST    https://{hostname|IP}/ngp/hypervisor
```

Parameters

Parameter 1: `action`

Start an ad hoc job.

- **Value:** `adhoc`
- **Type:** System string. Required. Available in the web user interface.

Data

Data 1: `slaPolicyName`

The name of the SLA policy.

- **Example value:** Gold
- **Type:** String. Required. Available in the web user interface.

Data 2: `subtype`

The type of the virtualized system, database, file system, cloud management, or containers.

- **Example value:** `vmware`
- **Type:** System string. Required. Available in the web user interface.

Tip: To get an {applicationName} value based on the name of the application name, follow the instructions in “Getting an {applicationName}” on page 222.

Data 3: `resource:`

The URLs of the instances to protect in the ad hoc backup job.

- **Value:**

A URL of a virtual machine uses the following syntax:

```
https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}
   /vm/{hypervisorVmId}
```

Tip: To get a {hypervisorVmId} by the virtual machine name, follow the instructions in “Getting a {hypervisorVmId} ” on page 229.

A URL of a database uses the following syntax:

```
https://{hostname|IP}/api/application/{applicationName}
   /instance/{applicationServerId}/database/{applicationDatabaseId}?from=flo
```

Example: Start an ad hoc backup for virtual machines

Assume that you have the following virtual machines hosted on different VMware vCenter servers:

- sales-americas.
  - Virtualized system: VMware - Dallas. Gold.
  - SLA policy: Gold
  - {hypervisorHypervisorId}: 1001
  - {hypervisorVmId}: ec44616651cda8b66fec4f7fd0daf554
- sales-apac
  - Virtualized system: VMware - Singapore
  - SLA policies: Gold and Diamond
  - {hypervisorHypervisorId}: 1002
  - {hypervisorVmId}: 2648550a670c6d383cba0d369ae4f91c

A Python snippet that is similar to the following example can be used to start an ad hoc backup job for these two virtual machines:

```
hypervisor_hypervisor_id_vm1 = "1001"    # VMware - Dallas
hypervisor_vm_id1 = "ec44616651cda8b66fec4f7fd0daf554"    # sales-americas
uri_vm1 = 'https://' + spp_ipv4 + '/api' \
    + '/hypervisor/' + hypervisor_hypervisor_id1 \
    + '/vm/' + hypervisor_vm_id1 \
    + '?from=hlo'

hypervisor_hypervisor_id_vm2 = "1002"    # VMware - Singapore
hypervisor_vm_id2 = "2648550a670c6d383cba0d369ae4f91c"    # sales-apac
uri_vm2 = 'https://' + spp_ipv4 + '/api' \
    + '/hypervisor/' + hypervisor_hypervisor_id2 \
    + '/vm/' + hypervisor_vm_id2 \
    + '?from=hlo'

_params = {"action": "adhoc"}

_data = f'''{{
    "slaPolicyName": "Gold",
    "subtype":       "vmware",
    "resource":      ["{uri_vm1}", "{uri_vm2}"]
}}'''

requests.post('https://' + spp_ipv4 + '/ngp/hypervisor/'
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK). Review the response to ensure that the backup job is running.

```
{
    "links": {...},
    "name": "vmware_Gold",
    "description": "Auto-generated job for Policy vmware_Gold",
    "policyId": "1045",
    "policyName": "vmware_Gold",
    "type": "protection",
    "typeDisplayName": "Protection",
    "subType": "vmware",
    "subTypeDisplayName": "VMware",
    "serviceId": "serviceprovider.protection.hypervisor",
    "displayName": "Hypervisor Backup",
    "status": "RESOURCE ACTIVE",
    "statusDisplayName": "Resource active",
```

```
        "lastSessionStatus": "FAILED",
        "lastSessionStatusDisplayName": "Failed",
        "triggerIds": [
            "1036"
        ],
        "triggerData": [
            {
                "triggerId": "1036",
                "triggerInfo": {
                    "subpolicy_id": "VADP Backup(2000)"
                }
            }
        ],
        "lastRunTime": 1580775121325,
        "nextFireTime": 1580775979000,
        "lastSessionDuration": 74949,
        "tenantId": 1000,
        "actions": null,
        "statistics": null,
        "policy": {
            "links": {...},
            "name": "vmware_Gold",
            "type": "protection",
            "subType": "vmware",
            "serviceId": "serviceprovider.protection.hypervisor",
            "description": "",
            "version": "3.0",
            "rbacPath": "root:0/policy:0/policytype:serviceprovider.protection.hyper
      visor/policy:1045",
            "tenantId": 1000,
            "creationTime": 1577725511302,
            "lastUpdated": 0,
            "spec": {
                "source": [],
                "storageworkflow": [
                    {
                        "id": "2000",
                        "name": "Gold",
                        "href": "http://localhost:8082/api/spec/storageprofile/2000"
                    }
                ],
                "option": {
                    "maxtasks": 0,
                    "vmsnapshot": {
                        "takesnapshot": false,
                        "concurrentsnapshotsonesx": 3,
                        "includememory": false,
                        "consistency": false,
                        "consistencyvms": [],
                        "scriptvms": [],
                        "truncateapplicationlogs": false
                    },
                    "skipreadonlydatastores": true
                },
                "notification": []
            },
            "logicalDelete": false,
            "script": {
                "preScript": null,
                "postScript": null,
                "preSnap": null,
                "postSnap": null,
                "preGuest": null,
                "postGuest": null,
                "preGuestSnap": null,
                "postGuestSnap": null,
                "continueScriptsOnError": false
            },
            "id": "1045"
        },
        "triggers": [
            {
                "links": {...},
                "name": "1038__VADP Backup(2000)",
                "type": "Schedule",
                "category": "INTERNAL",
                "triggerInfo": {
                    "id": 1036,
                    "name": "1036:1038",
                    "comment": null,
                    "creatorId": 0,
                    "creationTime": 1577725512115,
```

```
                "type": null,
                "flags": 0,
                "pk": null,
                "triggerId": "1036",
                "jobId": "1038",
                "triggerInfo": {
                    "subpolicy_id": "VADP Backup(2000)"
                },
                "category": "INTERNAL"
            },
            "properties": {
                "activateDate": 1564604779173,
                "deactivateDate": null,
                "timezone": null,
                "frequency": 4,
                "type": "HOURLY",
                "dowList": [
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false
                ],
                "domList": [
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false,
                    false
                ]
            },
            "rbacPath": "trigger:0/trigger:1036",
            "onHold": false,
            "id": "1036"
        }
    ],
    "lastrun": {
        "sessionId": "1580775119061",
        "jobName": "vmware_Gold",
        "type": "protection",
        "subType": "vmware",
        "serviceId": "serviceprovider.protection.hypervisor",
        "start": 1580775121325,
        "end": 1580775196274,
        "duration": 74949,
        "status": "FAILED",
        "results": null,
        "properties": {
            "statistics": [
                {
                    "resourceType": "vm",
                    "total": 2,
```

```
                        "success": 0,
                        "failed": 2,
                        "skipped": 0,
                        "names": null,
                        "jobSessionId": 0,
                        "policyId": null,
                        "startTime": 0
                    },
                    {
                        "resourceType": "datastore",
                        "total": 1,
                        "success": 0,
                        "failed": 0,
                        "skipped": null,
                        "names": null,
                        "jobSessionId": 0,
                        "policyId": null,
                        "startTime": 0
                    }
                ]
            },
            "numTasks": 2,
            "previousLastTask": 0,
            "lastUpdate": 1580775196299,
            "percent": 60,
            "policySnapshot": null
        },
        "id": "1038"
    }
```

Chapter 4. Manage Protection

You can complete operations to manage data protection. In the web user interface, most operations can be completed from the pages under **Manage Protection**.

```
reference_sla
refmanageprotection_virtualizedsystems
refmanageprotection_appservers
refmanageprotection_databases
reference_sql
```

Policy Overview

Service level agreement (SLA) policies, also known as backup policies, define parameters for backup jobs. These parameters include the frequency and retention period of backups and the option to replicate or copy backup data. You can use predefined SLA policies, or customize them to meet your needs.

You can complete operations with the REST API, as described in the following sections.

Getting information about SLA policies

You can get information about SLA policies.

Method and URI

To get information about all SLA policies, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/spec/storageprofile
```

To get information about a specific SLA policy, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/spec/storageprofile/{slapolicyId}
```

Tip: To get an {slapolicyId} value based on the name of the SLA policy, follow the information in “Getting an {slapolicyId}” on page 241.

Parameters

None.

Data

None.

Example 1: Get information about all SLA policies

A Python snippet that is similar to the following example can be used to request a list of all SLA policies and their information.

```
requests.get('https://' + spp_ipv4 + '/api/spec/storageprofile',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
```

```
 "total": 7,
 "slapolicies": [
     {
         "id": "2000",
         "links": {...},
         "description": "Gold SLA (hourly) to Primary site",
         "name": "Gold",
         "type": "backup",
         "version": "1.0",
         "spec": {
             "simple": true,
             "subpolicy": [
                 {
                     "type": "REPLICATION",
                     "retention": {
                         "age": 2
                     },
                     "useEncryption": false,
                     "software": true,
                     "trigger": {
                         "frequency": 4,
                         "type": "HOURLY",
                         "activateDate": 1564604760000,
                         "timezone": "America/Chicago"
                     },
                     "site": "Primary"
                 }
             ]
         },
         "rbacPath": "root:0/storageprofile:0/storageprofiletype:backup/storag
eprofile:2000"
     },
     {
         "id": "2001",
         "links": {...},
         "description": "Silver SLA (daily) to Primary site",
         "name": "Silver",
         ...
     },
     {
         "id": "2002",
         "links": {...},
         "description": "Bronze SLA (weekly) to Primary site",
         "name": "Bronze",
         ...,
     },
     {
         "id": "2004",
         "links": {...},
         "description": "Container SLA (daily) to Primary site",
         "name": "Container",
         "type": "snapshot_backup",
         ...,
     },
     {
         "id": "2005",
         "links": {...},
         "description": "EC2 SLA",
         "name": "EC2",
         "type": "snapshot",
         ...,
     },
     {
         "id": "2110",
         "links": {...},
         "description": null,
         "name": "Diamond",
         "type": "backup",
         ...,
     },
     {
         "id": "2111",
         "links": {...},
         "description": null,
         "name": "Sapphire",
         "type": "snapshot_backup",
         ...,
     },
     {
         "id": "2112",
         "links": {...},
         "description": null,
```

```
            "name": "Emerald",
            "type": "snapshot",
            ...,
        }
    ]
}
```

Example 2: Get information about a specific SLA policy

Assume that you added an SLA policy, Sapphire ({slapolicyId} 2111) to IBM Spectrum Protect Plus.

A Python snippet that is similar to the following example can be used to request information about this SLA policy:

```
sla_id = "2111"

requests.get('https://' + spp_ipv4 + '/api/spec/storageprofile/' + sla_id,
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "id": "2111",
    "links": {...},
    "description": null,
    "name": "Sapphire",
    "type": "snapshot_backup",
    "version": "1.0",
    "spec": {
        "simple": true,
        "subpolicy": [
            {
                "type": "SNAPSHOT",
                "retention": {
                    "age": 15
                },
                "trigger": {
                    "frequency": 1,
                    "type": "DAILY",
                    "activateDate": 1588982400000,
                    "timezone": "America/Chicago"
                },
                "label": "Sales Kubernetes"
            },
            {
                "type": "REPLICATION",
                "software": false,
                "retention": {},
                "useEncryption": false,
                "trigger": {
                    "frequency": 1,
                    "type": "DAILY",
                    "activateDate": 1588986000000,
                    "timezone": "Asia/Singapore"
                },
                "site": "Primary"
            },
            {
                "type": "SPPOFFLOAD",
                "retention": {},
                "target": {
                    "href": "https://10.0.0.100/api/cloud/2",
                    "resourceType": "cos",
                    "id": "2",
                    "wormProtected": false
                },
                "trigger": {
                    "frequency": 1,
                    "type": "DAILY",
                    "activateDate": 1589014800000,
                    "timezone": "America/Chicago"
                },
                "source": "backup"
            },
            {
```

```
                    "type": "SPPARCHIVE",
                    "retention": {
                        "age": 90
                    },
                    "target": {
                        "href": "https://10.0.0.100/api/cloud/4",
                        "resourceType": "aws",
                        "id": "4",
                        "wormProtected": false
                    },
                    "trigger": {
                        "frequency": 1,
                        "type": "WEEKLY",
                        "dowList": [
                            false,
                            true,
                            false,
                            false,
                            false,
                            false,
                            false,
                            false
                        ],
                        "activateDate": 1589018400000,
                        "timezone": "America/Chicago"
                    },
                    "source": "backup"
                },
                {
                    "type": "REPLICATION",
                    "retention": {
                        "age": 15
                    },
                    "useEncryption": false,
                    "software": true,
                    "trigger": {
                        "frequency": 1,
                        "type": "DAILY",
                        "activateDate": 1589007600000,
                        "timezone": "America/Chicago"
                    },
                    "site": "Dallas"
                }
            ]
        },
        "rbacPath": "root:0/storageprofile:0/storageprofiletype:snapshot_backup/stor
       ageprofile:2111"
    }
```

Adding an SLA policy

You can add an SLA policy by using the REST API. There are three types of SLA policies. Use one of the following types based on the backup target objects:

- Type 1: VMware, Hyper-V, Exchange, Microsoft 365, SQL, Oracle, Db2, MongoDB, IBM Spectrum Protect Plus Catalog, and Windows File Systems
- Type 2: Kubernetes, OpenShift
- Type 3: Amazon EC2

Method and URI

To add an SLA policy, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/api/spec/storageprofile
```

Parameters

None.

Data

Data 1: `name`

The name of the SLA policy.

- **Example value:** Diamond
- **Type:** String. Required. Available in the web user interface.

Data 2: `type`

The type of the SLA policies.

- **Value:** Use one of the following values:

Backup targets	Value
VMware, Hyper-V, Exchange, Microsoft 365, SQL, Oracle, Db2, MongoDB, Catalog, and Windows File Systems (Type 1)	`backup`
Kubernetes, OpenShift (Type 2)	`snapshot_backup`
Amazon EC2 (Type 3)	`snapshot`

- **Type:** System string. Required. Available in the web user interface.

Data 2: `spec`

Configuration settings for the SLA policy.

- **Value:** `{"subpolicy": ...}`
- **Type:** JSON object. Required. Available in the web user interface.

Data 2.1: `spec` > `subpolicy`

Configuration settings for the SLA policy by topics: Snapshot protection, a backup policy , a replication policy, making incremental copies to standard object storage, and making full copies to archive object storage.

- **Example value 1:** Define configuration settings in a JSON object for each topic. Some topics are optional. For example, assume that you want to make an SLA policy for VMware instances and Db2 instances (Type 1); and make the SLA policy has only the backup policy topic. In this case, the `subpolicy` value has the following structure:

```
[
    {
        "type":     "SNAPSHOT",
        ...
    }
]
```

- **Example value 2:** To create another SLA policy for Kubernetes or OpenShift instances (Type 2), and define configuration settings for all topics, the `subpolicy` value has the following structure:

```
[
    {
        "type":     "SNAPSHOT",
        ...
    },
    {
        "type":     "REPLICATION",
        "software": true,
```

```
            ...
        },
        {
            "type":     "REPLICATION",
            "software": false,
            ...
        },
        {
            "type":     "SPPOFLOAD",
            ...
        },
        {
            "type":     "SPPARCHIVE",
            ...
        }
    ]
```

- **Type:** Array. Required. Available in the web user interface.

Data 2.1.1: `spec` > `subpolicy` > `type`

The topics of the configuration settings. Available topics differ by the type of SLA policies.

- **Value:** Use one of the following values:

Topic	Value	Type 1	Type 2	Type 3
Snapshot protection	`SNAPSHOT`	Not available	Required	Required
Backup policy	`REPLICATION` with software	Required	Optional	Not available
Replication policy	`REPLICATION` without software	Optional	Optional	Not available
Incremental copy to standard object storage	`SPPOFFLOAD`	Optional	Optional	Not available
Full copy to archive object storage	`SPPARCHIVE`	Optional	Optional	Not available

When the value `REPLICATION` is used, you must define the Boolean value `software` which is described next.

- **Type:** System string. Required. Available in the browser user interface.

Data 2.1.2: `spec` > `subpolicy` > `software`

When the value `REPLICATION` is used for the Data key-value pair, `spec` > `subpolicy` > `type`, you must define the Boolean value `software`.

If `software` is true, the backup policy topic is used for the `subpolicy` JSON object. If the value of software is false, the replication policy topic is used.

- **Type:** Boolean. Available in the web user interface.

Data 2.1.3: `spec` > `subpolicy` > `retention`

The period to retain backup snapshots. Any backup snapshots that exceed the retention period will be purged.

Attention:

Be careful about the retention period when you edit SLA policies. If you update an SLA policy with a shorter retention period, IBM Spectrum Protect Plus purges all backup snapshots that are older than that retention period.

For example, assume that you add an SLA policy with the retention period of 90 days to make a backup snapshot every day. Use this policy for 3 months. Update it with the new retention period

of 3 days. Then, IBM Spectrum Protect Plus deletes 87 daily backup snapshots immediately and retains 3 snapshots that were created in the past 3 days.

- **Example value:** The following `retention` value can be used to retain backup snapshots for 365 days (1 year).

```
{
    "age": 365
}
```

- **Type:** JSON object. Required. Available in the web user interface.

Data 2.1.3.1: `spec` > `subpolicy` > `retention` > `age`

The number of days to retain backup snapshots.

- **Type:** Integer. Required. Available in the web user interface.

Data 2.1.4: `spec` > `subpolicy` > `useEncryption`

An indication of whether using only encrypted disk storage on external vSnap servers. This key-value pair is available for the backup policy topic only.

- **Example value:** `false` to allow IBM Spectrum Protect Plus to store backup snapshots on vSnap servers with unencrypted disks.
- **Type:** Boolean. Required. Available in the web user interface.

Data 2.1.5: `spec` > `subpolicy` > `trigger`

Define the frequency, start date and time, and the timezone for the web user interface.

- **Example value 1:** The following `trigger` value can be used to make a backup snapshot every 15 minutes, start the SLA policy from 1590987600 UNIX time (June 1, 2020 at 0:00 AM CDT); and the timezone for the web user interface is set to CDT:

```
{
    "frequency":    15,
    "type":         "SUBHOURLY",
    "activateDate": 1590987600000,
    "timezone":     "America/Chicago"
}
```

- **Example value 2:** The following `trigger` value is the same as Example value 1, except the frequency being set to one time every two hours:

```
{
    "frequency":    2,
    "type":         "HOURLY",
    "activateDate": 1590987600000,
    "timezone":     "America/Chicago"
}
```

- **Example value 3:** The following `trigger` value is the same as Example value 1, except the frequency being set to one time every two days:

```
{
    "frequency":    2,
    "type":         "DAILY",
    "activateDate": 1590987600000,
    "timezone":     "America/Chicago"
}
```

- **Example value 4:** The following `trigger` value is the same as Example value 1, except the frequency being set to one time every Sunday and Friday:

```
{
    "frequency":    1,
    "type":         "WEEKLY",
    "dowList": [ false,
        true,  false, false, false, false, true,  false
    ],
    "activateDate": 1590987600000,
    "timezone":     "America/Chicago"
}
```

- **Example value 5:** The following `trigger` value is the same as Example value 1, except the frequency being set to one time on the 1st day and on the 15th day every other month:

```
{
    "frequency":    2,
    "type":         "MONTHLY",
    "domList":      [ false,
        true,  false, false, false, false, false, false,
        false, false, false, false, false, false, false,
        true,  false, false, false, false, false, false,
        false, false, false, false, false, false, false,
        false, false, false, false
    ],
    "activateDate": 1590987600000,
    "timezone":     "America/Chicago"
}
```

- **Example value 6:** The following `trigger` value is the same as Example value 1, except the frequency being set to one time every three years, and the timezone for the web user interface being set to UTC:

```
{
    "frequency":    3,
    "type":         "YEARLY",
    "activateDate": 1590987600000,
    "timezone":     "UTC"
}
```

- **Type:** JSON object. Required. Available in the web user interface.

Data 2.1.5.1: `spec` > `subpolicy` > `trigger` > `frequency`

A number of unit periods that are defined by the `type` value to run a job that are associated with the SLA policy. The range of the number varies by the `type` value:

type	Minimum frequency	Maximum frequency	Description
SUBHOURLY	5	60	Between 5 minutes and 60 minutes
HOURLY	1	24	Between 1 hour and 24 hours
DAILY	15	30	Between 1 day and 30 days
WEEKLY	1	4	Between 1 week and 4 weeks
MONTHLY	1	60	Between 1 month and 12 months
YEARLY	15	60	Between 1 year and 60 years

- **Type:** Integer. Required. Available in the web user interface.

Data 2.1.5.2: `spec` > `subpolicy` > `trigger` > `type`

The time unit.

- **Value:** Use one of the following values for the time unit you want to use:

Time unit	Value
Minute	`SUBHOURLY`
Hour	`HOURLY`
Day	`DAILY`
Week	`WEEKLY`
Month	`MONTHLY`
Year	`YEARLY`

- **Type:** System string. Required. Available in the web user interface.

Data 2.1.5.3: `spec` > `subpolicy` > `trigger` > `dowList`

The days in a week to trigger the SLA policy. Use this key-value pair when you use `WEEKLY` for `type`.

The `dowList` is a list that contains 8 Boolean objects. The item in index 0 is reserved for the system, and it must be `false`. The item in index 1 is Sunday, the item in index 2 is Monday, and so on.

- **Example:** You can make multiple days to trigger the SLA policy. For example, to trigger every Sunday and Saturday, use the following list:

```
[
    false,
    true,  false, false, false, false, false, true
]
```

- **Restriction:** In the web user interface, you can define only one day in a week.
- **Type:** System list (length 8, Boolean). Partially available in the web user interface.

Data 2.1.5.4: `spec` > `subpolicy` > `trigger` > `domList`

The days in a month to trigger the SLA policy. Use this key-value pair when you use `MONTHLY` for `type`.

The `domList` is a list that contains 33 Boolean objects. The item in index 0 is reserved for the system, and it must be `false`. The item in index 1 is the 1st day of a month, the item in index 2 is the 2nd day of a month, and so on. The item in index 33 is the last day of a month.

Tip: It is advised not to use the items between index 28 and 31 because some months do not have days between 28 and 31.

Example value: The following list can be used to select the last day of every month (for example, January 31, February 28, and so on):

```
[
    false,
    false, false, false, false, false, false, false,
    false, false, false, false, false, false, false,
    false, false, false, false, false, false, false,
    false, false, false, false, false, false, false,
    false, false, false, true
]
```

Restriction: In the web user interface, you can define only one day in a month.

- **Type:** System list (length 33, Boolean). Partially available in the web user interface.

Data 2.1.5.5: spec > subpolicy > trigger > activateDate

The start time to activate the SLA policy. Use the UNIX timestamp format followed by three digits, which represent milliseconds.

- **Type:** Integer. Required. Available in the web user interface.

Data 2.1.5.6: spec > subpolicy > trigger > timezone

The Time Zone database name for the web user interface. This value does not affect the activateDate value in the UNIX time format for the start time. You can use UTC too.

Tip: Use UTC for UTC. To get a full list of TZ database names, see the web user interface or log in to IBM Spectrum Protect Plus via SSH and run the timedatectl list-timezones command.

- **Type:** System string. Required. Available in the web user interface.

Data 2.1.6: spec > subpolicy > source

The source for the incremental copy to standard object storage topic and the full copy to archive object storage topic. Use one of the following values:

Source	Value
Backup policy destination	backup
Replication policy destination	replication

- **Type:** System string. Required. Available in the web user interface.

Data 2.1.7: spec > subpolicy > site

The name of the site. This key-value pair is only available for the snapshot protection topic, the backup policy topic, and the replication policy topic.

- **Example value:** Primary
- **Type:** String. Required. Available in the web user interface.

Data 2.1.8: spec > subpolicy > target

Specify the object storage server or the repository server to store the backup snapshots. This key-value pair is only available for the incremental copy to standard object storage topic and the full copy to archive object storage topic.

- **Example value:**

```
{
    "href":         "https://10.0.0.100/api/cloud/3",
    "resourceType": "cos",
    "id":           "3"
}
```

- **Type:** JSON object. Required. Available in the web user interface.

Data 2.1.8.1: spec > subpolicy > target > href

The {cloudProviderInfoHref} of the object storage server or the repository server.

Tip: To get a {cloudProviderInfoHref} by the name of the storage server or the repository server, follow the instructions in “Getting a {cloudProviderInfoHref}” on page 225.

- **Type:** String. Required.

Data 2.1.8.2: spec > subpolicy > target > resourceType

The type of the object storage server or the repository server.

- **Value:** Use one of the following values:

Provider	Value
Amazon Simple Storage Service (S3)	`aws`
IBM Cloud Object Storage	`cos`
Microsoft Azure Blob Storage	`azure`
S3 compatible object storage	`generic`
IBM Spectrum Protect	`sp`

- **Type:** System string. Required.

Data 2.1.8.3: `spec` > `subpolicy` > `target` > `id`

The `{cloudProviderInfoId}` of the object storage server or the repository server.

Tip: To get a {cloudProviderInfoId} by the name of the storage server or the repository server, follow the instructions in "Getting a {cloudProviderInfoId}" on page 225.

- **Type:** String. Required.

Data 2.1.9: `spec` > `subpolicy` > `label`

The prefix for snapshot names. This key-value pair is available in SLA policies for Kubernetes and OpenShift (Type 2), and Amazon EC2 (Type 3).

Restriction: SLA policies for Kubernetes and OpenShift (Type 2) support only lowercase characters (a-z) or digits (0-9). SLA policies for Amazon EC2 does not have this restriction.

- **Example value:** daily
- **Type:** String. Available in the web user interface.

Example 1: Add an SLA policy for VMware instances (Type 1)

Assume that you want to add an SLA policy with the following properties:

- **Name:** Diamond
- **Type:** VMware, Hyper-V, Exchange, Microsoft 365, SQL Server, Oracle Database, Db2, MongoDB, IBM Spectrum Protect Plus Catalog, and Windows File Systems (Type 1)
- **Backup policy:** Required
  - **Retention:** 7 days
  - **Frequency:** every 2 hours
  - **Start time:** 1 June 2020 at 0:00 US Central Time (1590987600 UNIX time)
  - **Target Site:** Dallas
  - **Only use encrypted disk storage:** No
- **Replication policy:** Enable
  - **Retention:** 30 days
  - **Frequency:** every 1 day
  - **Start time:** 1 July 2020 at 13:00 Singapore Time (1593579600 UNIX time)
  - **Target Site:** Singapore
  - **Only use encrypted disk storage:** No
- **Incremental copy to standard object storage:** Enable
  - **Retention:** 7 days
  - **Frequency:** every 1 week, every Sunday
  - **Start time:** June 10, 2020 at 0:00 US Central Time (1591765200 UNIX time)

  - **Source:** Backup policy destination
    - **Destination and target:** Cloud services, BlueMachines IBM COS - Dallas (`{cloudProviderInfoId} 2`)
- **Full copy to archive object storage:** Enable
  - **Retention:** 5 years
  - **Frequency:** every 1 month, the 20th day of every month
  - **Start time:** 20 June 2020 at 2:15 US Central Time (1592637300 UNIX time)
  - **Source:** Backup policy destination
    - **Destination and target:** Repository Server: BlueMachines Spectrum Protect - Dallas 1 (`{cloudProviderInfoId} 6`)

A Python snippet that is similar to the following example can be used to create an SLA policy:

```
_data = f'''{{
   "name": "Diamond",
   "type": "backup",
   "spec": {{
      "subpolicy": [
         {{
            "type":          "REPLICATION",
            "software":      true,
            "retention":     {{
               "age": 7
            }},
            "useEncryption": false,
            "trigger":       {{
               "frequency":    2,
               "type":         "HOURLY",
               "activateDate": 1590987600000,
               "timezone":     "America/Chicago"
            }},
            "site":          "Dallas"
         }},
         {{
            "type":          "REPLICATION",
            "software":      false,
            "retention":     {{
               "age": 30
            }},
            "useEncryption": false,
            "trigger":       {{
               "frequency":    1,
               "type":         "DAILY",
               "activateDate": 1593579600000,
               "timezone":     "Asia/Singapore"
            }},
            "site":"Primary"
         }},
         {{
            "type":      "SPPOFFLOAD",
            "retention": {{
                "age": 7
            }},
            "trigger":   {{
               "frequency":    1,
               "type":         "WEEKLY",
               "dowList":      [ false,
                   true, false, false, false, false, false, false
               ],
               "activateDate": 1591765200000,
               "timezone":     "America/Chicago"
            }},
            "source":    "backup",
            "target":    {{
               "href":         "https://10.0.0.100/api/cloud/2",
               "resourceType": "cos",
               "id":           "2"
            }}
         }},
         {{
            "type":      "SPPARCHIVE",
            "retention": {{
```

```
                "age": 1825
            }},
            "trigger":   {{
                "frequency":    1,
                "type":         "WEEKLY",
                "dowList":      [ false,
                    true, false, false, false, false, false, false
                ],
                "activateDate": 1592637300000,
                "timezone":     "America/Chicago"
            }},
            "source":    "backup",
            "target":    {{
                "href":         "https://10.0.0.100/api/cloud/6",
                "resourceType": "sp",
                "id":           "6"
            }}
        }}
    ]
  }}
}}'''

requests.post('https://' + spp_ipv4 + '/api/spec/storageprofile',
    headers={...}, data=_data, verify=...)
```

The request prompts a response with the HTTP status of 201 (Created). The response body is similar to the JSON object for the SLA policy, Diamond, as shown in “Getting information about SLA policies” on page 45. Review the response to ensure that the new SLA policy, Diamond, was created.

Example 2: Add an SLA policy for Kubernetes instances (Type 2)

Assume that you want to add an SLA policy with the following properties:

- **Name:** Sapphire
- **Type:** Kubernetes (Type 2)
- **Snapshot protection:** Required
  - **Retention:** 7 days
  - **Frequency:** every 30 minutes
  - **Start time:** 1 June 2020 at 0:00 US Central Time (1589778000 UNIX time)
  - **Snapshot prefix:** sapphire
- **Backup policy:** Enable
  - **Retention:** 1 month
  - **Frequency:** every 1 day
  - **Start time:** 1 June 2020 at 1:00 US Central Time (1589778000 UNIX time)
  - **Target Site:** Dallas
  - **Only use encrypted disk storage:** No
- **Replication policy:** Enable
  - **Retention:** 30 days
  - **Frequency:** every 1 day
  - **Start time:** 1 July 2020 at 13:00 Singapore Time (1593579600 UNIX time)
  - **Target Site:** Singapore
  - **Only use encrypted disk storage:** No
- **Incremental copy to standard object storage:** Yes
  - **Retention:** 1 month
  - **Frequency:** every 1 day
  - **Start time:** June 10, 2020 at 0:00 US Central Time (1591765200 UNIX time)
  - **Source:** Replication policy destination

    - **Destination and target:** Cloud services, BlueMachines Amazon S3 US-East1 (`{cloudProviderInfoId}` 4)
- **Full copy to archive object storage:** Yes
  - **Retention:** 5 years
  - **Frequency:** every 1 month, the 20th day of every month
  - **Start time:** 20 June 2020 at 2:15 US Central Time (1592637300 UNIX time)
  - **Source:** Backup policy destination
    - **Destination and target:** Cloud services: BlueMachines Azure US-East (`{cloudProviderInfoId}` 3)

A Python snippet that is similar to the following example can be used to create an SLA policy:

```
_data = f'''{{
    "name": "Sapphire",
    "type": "snapshot_backup",
    "spec": {{
        "subpolicy": [
            {{
                "name":      "Storage Snapshot",
                "type":      "SNAPSHOT",
                "retention": {{
                    "age": 7
                }},
                "trigger":   {{
                    "frequency":    30,
                    "type":         "SUBHOURLY",
                    "activateDate": 1589778000000,
                    "timezone":     "America/Chicago"
                }},
                "label":     "sapphire"
            }},
            {{
                "type":          "REPLICATION",
                "software":      true,
                "retention":     {{
                    "age": 30
                }},
                "useEncryption": false,
                "trigger": {{
                    "frequency":    1,
                    "type":         "DAILY",
                    "activateDate": 1589778000000,
                    "timezone":     "America/Chicago"
                }},
                "site":          "Dallas"
            }},
            {{
                "type":          "REPLICATION",
                "retention":     {{
                    "age": 30
                }},
                "useEncryption": false,
                "software":      false,
                "trigger": {{
                    "frequency":    1,
                    "type":        "DAILY",
                    "activateDate": 1593579600000,
                    "timezone":     "Asia/Singapore"
                }},
                "site":          "Primary"
            }},
            {{
                "type":         "SPPOFFLOAD",
                "retention":    {{
                    "age": 30
                }},
                "trigger":      {{
                    "frequency":    1,
                    "type":         "DAILY",
                    "activateDate": 1591765200000,
                    "timezone": "America/Chicago"
                }},
                "source":       "replication",
```

```
                "target":       {{
                    "href":         "https://10.0.0.100/api/cloud/4",
                    "resourceType": "aws",
                    "id":           "4"
                }}
            }},
            {{
                "type":            "SPPARCHIVE",
                "retention":        {{
                    "age": 1825
                }},
                "trigger":          {{
                    "frequency": 1,
                    "type":      "MONTHLY",
                    "domList":   [ false,
                        false, false, false, false, false,
                        false, false, false, false, false,
                        false, false, false, false, false,
                        false, false, false, false, true,
                        false, false, false, false, false,
                        false, false, false, false, false,
                        false, false
                    ],
                    "activateDate": 1592637300000,
                    "timezone":     "America/Chicago"
                }},
                "source":           "replication",
                "target": {{
                    "href":         "https://10.0.0.100/api/cloud/3",
                    "resourceType": "azure",
                    "id":           "3"
                }}
            }}
        ]
    }}
}}'''

requests.post('https://' + spp_ipv4 + '/api/spec/storageprofile',
    headers={...}, data=_data, verify=...)
```

The request prompts a response with the HTTP status of 201 (Created). The response body is similar to the JSON object for the SLA policy, Sapphire, as shown in “Getting information about SLA policies” on page 45. Review the response to ensure that the new SLA policy, Sapphire, was created.

Example 3: Add an SLA policy for Amazon EC2 instances (Type 3)

Assume that you want to add an SLA policy with the following properties:

- **Name:** Emerald
- **Type:** Amazon EC2 (Type 3)
- **Snapshot protection:** Required
  - **Retention:** 6 months
  - **Frequency:** every 2 weeks on Sunday
  - **Start time:** 25 June 2020 at 0:00 US Central Time
  - **Snapshot prefix:** Emerald:

A Python snippet that is similar to the following example can be used to create an SLA policy:

```
_data = f'''{{
   "name": "Emerald",
   "type": "snapshot",
   "spec": {{
      "subpolicy": [
         {{
            "type":      "SNAPSHOT",
            "retention": {{
               "age": 180
            }},
            "trigger":   {{
               "frequency":    2,
               "type":         "WEEKLY",
```

```
                  "dowList":      [ false,
                     true,  false, false, false, false, false, false
                  ],
                  "activateDate": 1593061200000,
                  "timezone":     "America/Chicago"
               }},
               "label":     "Emerald:"
            }}
         ]
      }}
   }}'''

   requests.post('https://' + spp_ipv4 + '/api/spec/storageprofile',
       headers={...}, data=_data, verify=...)
```

The request prompts a response with the HTTP status of 201 (Created). The response body is similar to the JSON object for the SLA policy, Emerald, as shown in “Getting information about SLA policies” on page 45. Review the response to ensure that the new SLA policy, Emerald, was created.

Editing the configuration of an SLA policy

You can edit settings for a site. You can change the site name and other options for the default Primary and Secondary sites.

Method and URI

To update the configuration of an SLA policy, use a PUT method and a URI:

```
PUT     https://{hostname|IPv4}/api/spec/storageprofile/{slapolicyId}
```

Tip: To get an {slapolicyId} value based on the name of the site, follow the information in “Getting an {slapolicyId}” on page 241.

Parameters

None.

Data

The data key-value pairs are the same as for an SLA policy. For more information, follow the instructions in “Adding an SLA policy” on page 48.

If you use the same `name` value for the SLA policy name, the same SLA policy is updated. If you use a different name, the SLA policy will be updated and a copy of the original SLA policy will be created automatically.

Use the same or greater `spec` > `subpolicy` > `retention` > `age` value for the number of days to retain the backup snapshots. If you update an SLA policy with a smaller number of the retention days, IBM Spectrum Protect Plus purges all backup snapshots that exceed the retention days.

Example: Edit the configuration of an SLA policy

Assume that you added an SLA policy Diamond (`{slapolicyId}` 2121) as described in the section “Example 1: Add an SLA policy for VMware instances (Type 1)” on page 55 . You want to update this SLA policy:

- **Name:** Moissanite
- **Type:** VMware, Hyper-V, Exchange, Microsoft 365, SQL Server, Oracle Database, Db2, MongoDB, IBM Spectrum Protect Plus Catalog, and Windows File Systems (Type 1)
- **Backup policy:** Required
  - **Retention:** 7 days
  - **Frequency:** every 2 hours

  - **Start time:** 1 June 2020 at 0:00 US Central Time
  - **Target Site:** Dallas
  - **Only use encrypted disk storage:** No
- **Replication policy:** Disable
- **Incremental copy to standard object storage:** Disable
- **Full copy to archive object storage:** Disable

A Python snippet that is similar to the following example can be used to update the configuration of an SLA policy:

```
slapolicy_id = "2121"    # Diamond

_data = f'''{{
    "name":         "Moissanite",
    "type":         "backup",
    "spec":         {{
        "subpolicy": [
            {{
                "type":           "REPLICATION",
                "software":       true,
                "retention":      {{
                    "age": 2
                }},
                "useEncryption": false,
                "trigger":        {{
                    "frequency":    15,
                    "type":         "SUBHOURLY",
                    "activateDate": 1590901200000,
                    "timezone":     "America/Chicago"
                }},
                "site":           "Dallas"
            }}
        ]
    }}
}}'''

requests.put('https://10.0.0.100/api/spec/storageprofile/' + slapolicy_id,
    headers={...}, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 200 (OK) and that the SLA policy is updated.

Deleting an SLA policy

You can delete a SLA policy.

Method and URI

To delete an SLA policy with {slapolicyId}, the request's method and URI are similar to the following example:

```
DELETE  https://{hostname|IPv4}/api/spec/storageprofile/{slapolicyId}
```

Tip: To get an {slapolicyId} value based on its name, follow the instructions in “Getting an {slapolicyId}” on page 241.

Parameters

Parameter 1: `name`

Specify the name of the SLA policy to delete.

- **Example value** `Diamond`
- **Type:** System string. Required. Available in the web user interface.

None.

Data

None.

Example: Delete an SLA policy

Assume that you have an SLA policy, Diamond, and you want to delete it.

A Python snippet that is similar to the following example can be used to delete the SLA policy:

```
sla_id = "2101"

requests.delete('https://' + spp_ipv4 + '/api/spec/storageprofile/' + sla_id,
    headers={...}, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 204 (No Content) and that you no longer see the SLA policy Diamond.

Managing resources assigned to an SLA policy

You can assign or unassign resources to an SLA policy.

Method and URI

To assign or unassign resources to an SLA policy with {slapolicyId}, the request method and URI are similar to the following examples:

Assign or unassign a hypervisor resource:

```
POST /ngp/hypervisor?action=applySLAPolicies
```

Assign or unassign an application resource:

```
POST /ngp/application?action=applySLAPolicies
```

Parameter

Parameter:

- **name:** `action`
- **value:** `applySLAPolicies`

Data

Data 1: `subtype`

The subtype of the resource.

- **Allowable values include:** `vmware, hyperv, awsec2, office365, db2, exchange, mongo, oracle, sql, catalog`

Data 2: `version`

- **Value:** `1.0`

Data 3: `resources`

A URL of a virtual machine that uses the following syntax:

```
https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}
   /vm/{hypervisorVmId}
```

Tip: To get a {hypervisorVmId} by the virtual machine name, follow the instructions in “Getting a {hypervisorVmId} ” on page 229.

A URL of a database that uses the following syntax:

```
https://{hostname|IP}/api/application/{applicationName}
   /instance/{applicationServerId}/database/{applicationDatabaseId}?from=flo
```

To see the resources of the specified subtype that are assigned to the SLA, see the following examples.

Example: Assign VMware virtual machines to an SLA policy

You can assign one or more VMware virtual machines to an SLA policy.

```
POST /ngp/hypervisor?action=applySLAPolicies

{
    "subtype": "vmware",
    "version": "1.0",
    "resources": [
        {
            "href": "https://10.11.61.28/api/hypervisor/1001/vm/
bbacb9bf43a9cac9b9f5fd083eb33433?from=hlo",
            "id": "bbacb9bf43a9cac9b9f5fd083eb33433",
            "metadataPath": "/MS:1001/DC:86ee95cb6cdb19cd8167a684279d8bec/
FLDR:a49910697066e5d9faeec4ada0ecfb16/FLDR:fcb0015cc595b53e7b3bc5bcd45b56f0/
FLDR:d15f2e70633bd9e85ed1513c8646c75d/VM:bbacb9bf43a9cac9b9f5fd083eb33433"
        }
    ],
    "slapolicies": [
        {
            "href": "https://10.11.61.28/api/spec/storageprofile/2102",
            "id": "2102",
            "name": "vsnapnosched"
        }
    ]
}
```

Example: Assign databases to an SLA policy

You can assign one or more databases to an SLA policy.

```
POST /ngp/application?action=applySLAPolicies

{
    "subtype": "sql",
    "version": "1.0",
    "resources": [
        {
            "href": "https://10.11.36.167/api/application/sql/instance/471030770/database/
ea8ba6d1888f6a7e0db04de18b5d7925?from=hlo",
            "id": "ea8ba6d1888f6a7e0db04de18b5d7925",
            "metadataPath": "/DBINST:471030770/DB:ea8ba6d1888f6a7e0db04de18b5d7925"
        }
    ],
    "slapolicies": [
        {
            "href": "https://10.11.36.167/api/spec/storageprofile/2002",
            "id": "2002",
            "name": "Bronze"
        }
    ]
}
```

Virtualized Systems (VMware, Hyper-V, Amazon EC2)

Some operations for virtualized systems share the same requests and response structure. For example, invoke an inventory job for virtualized systems and it takes place for all types of virtualized systems, namely, VMware, Microsoft Hyper-V, and Amazon Elastic Compute Cloud (EC2). This section describes operations that are commonly used by multiple virtualized systems.

Running an inventory job for virtualized systems

An inventory job for virtualized systems will rebuild a list of those and resources on it, such as virtual machines and virtual hard disks on it, that IBM Spectrum Protect Plus can protect. This section discusses how you can invoke an inventory job for hypervisors that applies to VMware vCenter, Microsoft Hyper-V, and Amazon EC2.

Every time you add a new hypervisor server or make any changes in their resources such as deploying new virtual machines on it, you have to run an inventory job at IBM Spectrum Protect Plus to update its list of hypervisor instances. To run an inventory job for application servers (Oracle Database, SQL Server, Db2, MongoDB, Exchange Server, Microsoft 365, Kubernetes, and OpenShift), follow the instructions in “Running an inventory job for application server instances” on page 82.

Method and URI

To start an inventory job, use a POST method and a URI:

```
POST    https://{hostname|IPv4}/api/endeavour/job/{jobId}
```

Tip: To get the {jobId} for the hypervisor inventory job, follow the instructions in “Getting a {jobId} ” on page 234, where the object name is `Hypervisor Inventory`. This {jobId} may vary by IBM Spectrum Protect Plus appliance deployments.

Parameters

Parameter 1: `action`

Specify the type of action.

- **Value:** `inventory`
- **Type:** System string. Required. Available in the web interface.

Parameter 2: `actionname`

Specify the step of the action.

- **Value:** `start`
- **Type:** System string. Required. Available in the web interface.

Data

None.

Example: Start an inventory job for virtualized system instances

Assume that you want to run a hypervisor inventory job for virtualized systems.

Create a function that converts the policy name, `Hypervisor Inventory`, into the job ID:

```
job_id = converter_job("Hypervisor Inventory")
print(job_id)
```

```
1003
```

Assume that the job ID is 1003 in this IBM Spectrum Protect Plus appliance. A Python snippet that is similar to the following example can be used to start the hypervisor inventory job:

```
_params = {
    "action":     "start",
    "actionname": "start"
}
```

```
_response = requests.post('https://' + spp_ipv4 + '/api/endeavour/job/' + job_id
    headers={...}, params=_params, verify=...)
```

You will receive a response with no body and the HTTP status code of 204 (No Content).

Attention: Do not determine the status of inventory jobs by referring to the HTTP status code in a response. This request only starts and inventory job and receives a response with the HTTP status code of 204 (No Content) regardless of the status or the result. To see the status of the inventory jobs, follow the information in “Getting the status of a hypervisor inventory job” on page 65.

Getting the status of a hypervisor inventory job

You can get status information about a currently running inventory job and the previous one.

Method and URI

To get information about the present and previous hypervisor inventory jobs, use a GET method and a URI, where `{jobId}` is the ID for the hypervisor inventory job:

```
GET     https://{hostname|IP}/endeavour/job/{jobId}
```

Parameters

None.

Example: Get status of a hypervisor inventory job

Assume that the `{jobId}` of the hypervisor inventory job is 1003.

A Python snippet that is similar to the following example can be used to get status information about the present and previous hypervisor inventory job:

```
job_id = "1003"  # Hypervisor Inventory

requests.get('https://' + spp_ipv4 + '/api/endeavour/job/' + job_id,
    headers={...}, verify=...)
```

Assume that you ran an inventory job for virtualized systems and send the request to see the status of it. Normally, the inventory job takes a few minutes to complete and so you should see the running status. In your browser, the blue **Run Inventory** button has been grayed out and reading **Inventory In Progress**. Here is an example response with the HTTP status code of 200 (OK).


```
{
    "links": {...},
    "name": "Hypervisor Inventory",
    "description": null,
    "policyId": "1003",
    "policyName": "Hypervisor Inventory",
    "type": "catalog",
    "typeDisplayName": "Catalog",
    "subType": "hypervisor",
    "subTypeDisplayName": "Hypervisor",
    "serviceId": "serviceprovider.catalog.hypervisor",
    "displayName": "Hypervisor Inventory",
    "status": "RUNNING",
    "statusDisplayName": "Running",
    "lastSessionStatus": "PARTIAL",
    "lastSessionStatusDisplayName": "Partial",
    "triggerIds": [
        "1003"
    ],
    "triggerData": [
        {
            "triggerId": "1003",
            "triggerInfo": null
```

```
            }
        ],
        "lastRunTime": 1578679221737,
        "nextFireTime": 1578729600000,
        "lastSessionDuration": 685498,
        "tenantId": 1000,
        "actions": null,
        "statistics": null,
        "lastrun": {
            "sessionId": "1578679221158",
            "jobName": "Hypervisor Inventory",
            "type": "catalog",
            "subType": "hypervisor",
            "serviceId": "serviceprovider.catalog.hypervisor",
            "start": 1578679221737,
            "end": 1578679907235,
            "duration": 685498,
            "status": "PARTIAL",
            "results": "Started",
            "properties": null,
            "numTasks": 4,
            "previousLastTask": 0,
            "lastUpdate": 1578679907240,
            "percent": 100,
            "policySnapshot": null
        },
        "id": "1003"
    }
```

In the example response above, the `status` value reads `RUNNING` which means the hypervisor inventory job was running at the time you sent the request. You can also see the information about the previous hypervisor inventory job.

A few minutes later, you send the same request again to see the status. At this time, the button in the browser comes back to the original, blue **Run Inventory** button. Here is an example response with the HTTP status code 200 (OK). As you see, the `status` value reads `IDLE`.

```
    {
        "links": {...},
        ...
        "status": "IDLE",
        "statusDisplayName": "Idle",
        "lastSessionStatus": "PARTIAL",
        "lastSessionStatusDisplayName": "Partial",
        "triggerIds": [
            "1003"
        ],
        "triggerData": [
            {
                "triggerId": "1003",
                "triggerInfo": null
            }
        ],
        "lastRunTime": 1578681060744,
        "nextFireTime": 1578729600000,
        "lastSessionDuration": 529178,
        "tenantId": 1000,
        "actions": null,
        "statistics": null,
        "lastrun": {...},
        ...
    }
```

Restriction: Do not determine the status of the hypervisor inventory jobs by referring to the HTTP status code in a response. This request will generate a response with the HTTP status code of 200 (OK) regardless of the status. Instead, refer to the relevant key-value pairs in the response to check the status.

Getting information about virtualized systems

You can get information about virtualized systems including vCenter Servers, Microsoft Hyper-V Servers, and Amazon EC2. The REST API can request a list of virtualized systems of all types, unlike the web user interface, which displays only one type.

Method and URI

To get information about all virtualized systems, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/hypervisor
```

To get information about all virtualized systems, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}
```

Tip: To get a {hypervisorHypervisorId} value based on its name, follow the instructions in “Getting a {hypervisorHypervisorId} ” on page 227.

Parameters

Parameter 1: `filter`

Optionally, you may use the filter to specify a type of virtualized systems. You can use the `filter` operation parameters that are described in “Filter” on page 243. If you do not use this parameter, you will get all types of virtualized systems.

- **Example value:** Use the following value for the `filter` key to get objects that are managed by VMware vCenter:

```
{
    [
        {
            "property": "type",
            "value":    "vmware",
            "op":       "="
        }
    ]
}
```

- **Type:** JSON object. Available in the web user interface.

Data

None.

Example 1: Get information about all virtualized systems

A Python snippet that is similar to the following example can be used to get a list of all virtualized systems:

```
requests.get('https://' + spp_ipv4 + '/api/hypervisor',
    headers={...}, verify=...)
```

The response below with the HTTP status of 200 (OK) shows the list of all virtualized systems.

```
{
    "links": {...},
    "total": 5,
    "hypervisors": [
        {
            "links": {...},
```

```
            "name": "10.0.0.10",
            "hostAddress": "10.0.0.10",
            "user": {
                "href": "https://10.0.0.100:-1/api/identity/user/2102"
            },
            "sslConnection": true,
            "portNumber": 443,
            "type": "vmware",
            "id": "1001",
            "uniqueId": "fa8f1a0f-9d01-44e6-b281-6c577cd920ae",
            "version": null,
            "build": null,
            "apiVersion": null,
            "properties": {},
            "logicalDelete": false,
            "opProperties": {
                "snapshotConcurrency": 3,
                "veServerInfo": null
            },
            "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:vmware/hypervis
or:1001",
            "resourceType": "hypervisor"
        },
        {
            "links": {...},
            "name": "hyperv.dallas.bluemachines.com.invalid",
            "hostAddress": "hyperv.dallas.bluemachines.com.invalid",
            "user": {
                "href": "https://10.0.0.100:-1/api/identity/user/2104"
            },
            "sslConnection": false,
            "portNumber": 5985,
            "type": "hyperv",
            "id": "1002",
            "uniqueId": "2d60ba76-9161-3492-ae5d-80f0b067760a",
            "version": null,
            "build": null,
            "apiVersion": null,
            "properties": {
                "sysinfoVersion": {
                    "key": "version",
                    "value": "10.0.17763"
                },
                "sysinfoModel": {
                    "key": "model",
                    "value": "PowerEdge R610"
                },
                "sysinfoManufacturer": {
                    "key": "manufacturer",
                    "value": "Dell Inc."
                },
                "sysinfoDomain": {
                    "key": "domain",
                    "value": "hyperv.dallas.bluemachines.com.invalid"
                },
                "sysinfoOs": {
                    "key": "os",
                    "value": "Microsoft Windows Server 2019 Datacenter"
                },
                "sysinfoName": {
                    "key": "name",
                    "value": "win2016enghv"
                }
            },
            "logicalDelete": false,
            "opProperties": {
                "snapshotConcurrency": 3,
                "veServerInfo": null
            },
            "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:hyperv/hypervis
or:1002",
            "resourceType": "hypervisor"
        },
        {
            "links": {...},
            "name": "us-east-1",
            "hostAddress": "10.0.0.100",
            "user": {
                "href": "https://10.0.0.100/api/identity/user/1002"
            },
            "sslConnection": false,
            "portNumber": 11001,
```

```
            "type": "awsec2",
            "id": "1003",
            "uniqueId": "us-east-1",
            "version": null,
            "build": null,
            "apiVersion": null,
            "properties": {},
            "logicalDelete": false,
            "accountName": "testEc2",
            "displayName": "us-east-1",
            "opProperties": {
                "snapshotConcurrency": 0,
                "veServerInfo": null
            },
            "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:awsec2/hypervis
   or:1005",
            "resourceType": "hypervisor"
        },
        {...}, {...}, ..., {...}
    ]
}
```

Example 2: Get information about a specific virtualized system

Assume that you want to get information about a specific virtualized system: 10.0.0.20. You get the hypervisorHypervisorId of it: 1005.

A Python snippet that is similar to the following example can be used to request information about this virtualized system:

```
hypervisor_hypervisor_id = "1005"

requests.get('https://' + spp_ipv4 + '/api/hypervisor/'
    + hypervisor_hypervisor_id,
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "name": "10.0.0.20",
    "hostAddress": "10.0.0.20",
    ...,
    "id": "1005",
    ...
}
```

Example 3: Get information about a specific type of virtualized systems

Assume that you only want to get information for vCenter Servers. Add the following parameter to the request, as shown in Example 1.

```
_params = {"filter": json.dumps([
              {"property": "type",
               "value":    "vmware",
               "op":       "="
              }
          ])}

requests.get('https://' + spp_ipv4 + '/api/hypervisor',
    headers={...}, param=_param, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "total": 2,
    "hypervisors": [
```

```
        {
            "links": {...},
            "name": "10.0.0.10",
            "hostAddress": "10.0.0.10",
            ...
            "type": "vmware",
            "id": "1001",
            ...
        },
        {
            "links": {...},
            "name": "10.0.0.20",
            "hostAddress": "10.0.0.20",
            ...,
            "id": "1005",
            ...
        }
    ]
}
```

Searching virtualized system instances

Search instances of virtualized systems.

Method and URI

To search VMware instances with a search string, use a GET method and a URI:

```
POST    https://{hostname|IP}/api/hypervisor/search
```

Parameters

Parameter 1: `resourceType`

The resource type to search with. In the web user interface, you can select one from the **View:** drop-down list.

- **Value:** Use one of the following value:

Search target	Value
Virtual machines and templates	vm
Datastore	volume
Tags	tag
Tags and categories	tagcategory

- **Type:** System string. Required. Available in the web user interface.

Parameter 2: `from`

- **Value:** `hlo`
- **Type:** System string. Required.

Parameter 3: `filter`

Optionally, you may use a filter to specify the search target. If this parameter is not used, IBM Spectrum Protect Plus searches within all virtual machines. You can use the `filter` operation parameters that are described in “Filter” on page 243.

- **Example value:** To search within instances that are associated with none of SLA policies, use the following filter parameter:

```
[
    {
        "property": "unprotected",
```

```
            "value":    "true",
            "op":       "="
        }
    ]
```

- **Example value:** To search with instances that are associated with the SLA policy, Gold, use the following parameter:

```
    [
        {
            "property": "storageProfileName",
            "value":    "Gold",
            "op":       "="
        }
    ]
```

- **Type:** Array. Available in the web user interface.

Data

Data 1: `name`

The search string. The `name` value can contain the following wildcard characters:

Wildcard character	Description
* (asterisk)	Matches zero or more characters.
? (question mark)	Matches any single character at the present position.

- **Example value:** `sales-*`
- **Type:** String. Required. Available in the web user interface.

Data 2: `hypervisorType`

The type of virtualized system servers.

- **Value:** Use one of the following value:

Virtualized system	Value
VMware vCenter	vmware
Hyper-V	hyperv
Amazon EC2	awsec2

- **Type:** String. Required. Available in the web user interface.

Example: Search VMware instances for virtual machines with a specific SLA policy

Assume that you have VMware assets in IBM Spectrum Protect Plus. You want to retrieve names of all virtual machines with the following properties:

- **Resource type:** Virtual machines and templates
- **Filter:**
  - **SLA policy:** Diamond
- **Hypervisor type:** VMware
- **Name:** `\*-dallas?`

The string with the wildcard characters represents the name beginning with zero or more of any characters, followed by -dallas, and ending with any single character.

```
_params = {
    "resourceType": "vm",
    "from": "hlo",
    "filter": {
        json.dumps([
            {
                "property": "storageProfileName",
                "value":    "Diamond",
                "op":       "="
            }
        ])
    }
}

_data = f'''{{
    "name": "*-dallas?",
    "hypervisorType": "vmware"
}}'''

requests.post('https://' + spp_ipv4 + '/api/hypervisor/search',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "total": 2,
    "vms": [
        {
            "links": {...},
            "name": "vsnap-dallas1",
            "type": "vm",
            "association": null,
            "protectionInfo": null,
            "recoveryInfo": null,
            "recoveryPoint": null,
            "siteId": null,
            "autoProvisionedsubPolicyName": null,
            "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:vmware/hypervis
   or:1004/vdc:717dafb25bbbca30d8dc72ed3cbf81f0/vfld:716779e50daefc126bbaed0ef3c36
   22d/vm:8848ad8c1d38ef880cb892a5570d50f5",
            "rbacPathId": 4146,
            "extInfo": null,
            "location": "/vmware-dallas1.bluemachines.com.invalid/Dallas - DataC
   enter1/Production folder",
            "metadataPath": "/MS:1004/DC:717dafb25bbbca30d8dc72ed3cbf81f0/FLDR:7
   16779e50daefc126bbaed0ef3c3622d/VM:8848ad8c1d38ef880cb892a5570d50f5",
            "inHLO": true,
            "hypervisorType": "vmware",
            "hypervisorKey": "1004",
            "hypervisorManagementServerID": "1004",
            "nativeKey": "vm-104786",
            "cloudType": null,
            "tags": [],
            "tagCategories": [],
            "metadataPathVmHost": "/MS:1004/DC:717dafb25bbbca30d8dc72ed3cbf81f0/
   CLUST:14100077601aaa13a1fdf555a065d58f/HOST:735cda83d7290ea7fa685ae7cdc3e95c/VM
   :8848ad8c1d38ef880cb892a5570d50f5",
            "rbacPathVmHost": "root:0/hypervisor.all:0/hypervisor.type:vmware/hy
   pervisor:1004/vdc:717dafb25bbbca30d8dc72ed3cbf81f0/vcluster:14100077601aaa13a1f
   df555a065d58f/vhost:735cda83d7290ea7fa685ae7cdc3e95c/vm:8848ad8c1d38ef880cb892a
   5570d50f5",
            "rbacPathIdVmHost": 4147,
            "alternatePaths": [
                {
                    "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:vmware/
   hypervisor:1004/vtagcategory:a6068ad43059b6b6fc3a0986ba7a0bed/vtag:a3269bb10cfb
   e7a1588a5a3a5b3637de/vm:8848ad8c1d38ef880cb892a5570d50f5",
                    "rbacPathId": 4148,
                    "metadataPath": "/MS:1004/TAGCATEGORY:a6068ad43059b6b6fc3a09
   86ba7a0bed/TAG:a3269bb10cfbe7a1588a5a3a5b3637de/VM:8848ad8c1d38ef880cb892a5570d
   50f5"
                }
            ],
```

```
            "powerSummary": {
                "powerState": "ON",
                "lastPowerOnTime": null,
                "lastPowerOffTime": null,
                "lastSuspendTime": null,
                "uptime": 1574371737324
            },
            "storageSummary": {
                "commited": 80351072860,
                "uncommited": 92048196830,
                "shared": 78085357568
            },
            "hypervisorHostKey": "735cda83d7290ea7fa685ae7cdc3e95c",
            "volumes": [
                {
                    "name": "IBM_Flash_System-Dallas",
                    "key": "00613cbaf57e385be9853046d07271f5",
                    "configVolume": true
                },
                {
                    "name": "isoImages",
                    "key": "ece8716493cd0d5b7283552b67bf263c",
                    "configVolume": false
                }
            ],
            "hypervisorFolderKey": "716779e50daefc126bbaed0ef3c3622d",
            "hypervisorFolderPath": "Production folder",
            "uniqueId": "4209f06b-f688-c4af-90d2-e93b6e40f270:5009d293-7abc-4b48
    -4a29-818f3045445e",
            "profileId": null,
            "keyId": null,
            "keyProviderId": null,
            "configFilePath": "[SG_STORAGE] sales-apac/sales-apac.vmx",
            "fileDirPath": null,
            "datacenter": {
                "name": "BlueMachines_vCenter_Dallas1",
                "key": "717dafb25bbbca30d8dc72ed3cbf81f0"
            },
            "cluster": {
                "name": "PS_SEcluster",
                "key": "14100077601aaa13a1fdf555a065d58f"
            },
            "vmGroup": null,
            "resourcePool": {
                "name": "Resources",
                "key": "784eb71414bd617fdb01be7f5d7acd50"
            },
            "cpu": 1,
            "coresPerCpu": 1,
            "memory": 2147483648,
            "vmVersion": "vmx-13",
            "customAttributes": {},
            "vmStorageUtilization": null,
            "configInfo": {
                "name": null,
                "type": null,
                "association": null,
                "protectionInfo": null,
                "recoveryInfo": null,
                "recoveryPoint": null,
                "siteId": null,
                "autoProvisionedsubPolicyName": null,
                "rbacPath": null,
                "rbacPathId": null,
                "extInfo": null,
                "location": null,
                "metadataPath": null,
                "inHLO": true,
                "hypervisorType": null,
                "hypervisorKey": null,
                "hypervisorManagementServerID": null,
                "nativeKey": null,
                "cloudType": null,
                "tags": null,
                "tagCategories": null,
                "metadataPathVmHost": null,
                "rbacPathVmHost": null,
                "rbacPathIdVmHost": null,
                "alternatePaths": null,
                "vmId": null,
                "osName": null,
                "osVersion": null,
```

```
                "hostName": null,
                "vmToolsStatus": "NOT_INSTALLED",
                "guestNetworkInfo": [],
                "windows": false,
                "linux": false,
                "nativeObject": {},
                "hypervisorHostKey": null,
                "volumeId": null,
                "volumes": [],
                "protected": false,
                "systemHold": false,
                "id": null
            },
            "attachments": [],
            "vmMetadata": {},
            "encrypted": false,
            "hypervisorHostname": "vmware-singapore1.bluemachines.com.invalid",
            "vmPoolDedupCompressionInfo": null,
            "windows": false,
            "template": false,
            "linux": false,
            "versionId": "vm.8848ad8c1d38ef880cb892a5570d50f5.1578038400234",
            "nativeObject": {},
            "volumeId": null,
            "protected": false,
            "systemHold": false,
            "id": "8848ad8c1d38ef880cb892a5570d50f5",
            "config": {
                "name": null,
                "type": null,
                "association": null,
                "protectionInfo": null,
                "recoveryInfo": null,
                "recoveryPoint": null,
                "siteId": null,
                "autoProvisionedsubPolicyName": null,
                "rbacPath": null,
                "rbacPathId": null,
                "extInfo": null,
                "location": null,
                "metadataPath": null,
                "inHLO": true,
                "hypervisorType": null,
                "hypervisorKey": null,
                "hypervisorManagementServerID": null,
                "nativeKey": null,
                "cloudType": null,
                "tags": null,
                "tagCategories": null,
                "metadataPathVmHost": null,
                "rbacPathVmHost": null,
                "rbacPathIdVmHost": null,
                "alternatePaths": null,
                "vmId": null,
                "osName": null,
                "osVersion": null,
                "hostName": null,
                "vmToolsStatus": "NOT_INSTALLED",
                "guestNetworkInfo": [],
                "windows": false,
                "linux": false,
                "nativeObject": {},
                "hypervisorHostKey": null,
                "volumeId": null,
                "volumes": [],
                "protected": false,
                "systemHold": false,
                "id": null
            },
            "resourceType": "vm",
            "storageProfiles": [],
            "copies": null
        },
        {
            "links": {...},
            "name": "oracle-dallas2",
            "type": "vm",
            "association": null,
            "protectionInfo": null,
            "recoveryInfo": null,
            "recoveryPoint": null,
            "siteId": null,
```

```
            "autoProvisionedsubPolicyName": null,
            "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:vmware/hypervis
    or:1004/vdc:717dafb25bbbca30d8dc72ed3cbf81f0/vfld:716779e50daefc126bbaed0ef3c36
    22d/vm:2648550a670c6d383cba0d369ae4f91c",
            "rbacPathId": 5310,
            "extInfo": null,
            "location": "/vmware-dallas1.bluemachines.com.invalid/Dallas - Datac
    enter1/Production folder",
            "metadataPath": "/MS:1004/DC:717dafb25bbbca30d8dc72ed3cbf81f0/FLDR:7
    16779e50daefc126bbaed0ef3c3622d/VM:2648550a670c6d383cba0d369ae4f91c",
            "inHLO": true,
            "hypervisorType": "vmware",
            "hypervisorKey": "1004",
            "hypervisorManagementServerID": "1004",
            "nativeKey": "vm-123790",
            "cloudType": null,
            "tags": [],
            "tagCategories": [],
            ...
        }
    ]
}
```

Getting virtualized system instances with view filters

You can use view filters to filter the information about virtualized system instances. Use one of the following view filters:

View filter	{hypervisorView}	VMware	Hyper-V	Amazon EC2
Tags	tagcontent	X		X
Datastore	storagecontent	X	X	
Hosts and clusters	hostcontent	X		

Method and URI

To get information about all virtualized systems with one of the view filters, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}/{hypervisorView}
```

Tip: To get a {hypervisorHypervisorId} based on the hostname or the IPv4 address, follow the information in “Getting a {hypervisorHypervisorId} ” on page 227.

Parameters

Parameter 1: `from`

- **Value:** `hlo`
- **Type:** System string. Required.

Data

None.

Example 1: Get information about virtualize system instances with the tag view filter

Assume that you added VMware vCenter Server: VMware - Dallas 1 ({`hypervisorHypervisorId`} 1001). A Python snippet that is similar to the following example can be used to request information about virtualized system instances with the tag view filter:

```
hypervisor_hypervisor_id = "1001"    # VMware - Dallas
```

```
_params = {"from": "hlo"}

requests.get('https://' + spp_ipv4 + '/api/hypervisor/'
    + hypervisor_hypervisor_id + '/tagcontent',
    headers={...}, params=_params, data="", verify=...)
```

The response below with the HTTP status of 200 (OK) shows the list of all virtualized systems.

```
{
    "links": {...},
    "total": 2,
    "contents": [
        {
            "links": {...},
            "name": "All",
            "type": "tagcategory",
            "association": null,
            "protectionInfo": null,
            "recoveryInfo": null,
            "recoveryPoint": null,
            "siteId": null,
            "autoProvisionedsubPolicyName": null,
            "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:vmware/hypervis
   or:1001/vtagcategory:512150d081814f93e4deb6055d66bcf4",
            "rbacPathId": 2,
            "extInfo": null,
            "location": "/DALLAS.BLUEMACHINES.com.invalid",
            "metadataPath": "/MS:1001/TAGCATEGORY:512150d081814f93e4deb6055d66bc
   f4",
            "inHLO": true,
            "hypervisorType": "vmware",
            "hypervisorKey": "1001",
            "hypervisorManagementServerID": "1001",
            "nativeKey": "urn:vmomi:InventoryServiceCategory:24b74365-f672-4aac-
   8862-0f733c5cf9de:GLOBAL",
            "cloudType": null,
            "tags": null,
            "tagCategories": null,
            "metadataPathVmHost": null,
            "rbacPathVmHost": null,
            "rbacPathIdVmHost": null,
            "alternatePaths": null,
            "description": "",
            "cardinality": "MULTIPLE",
            "associableTypes": [],
            "nativeObject": {},
            "hypervisorHostKey": null,
            "windows": false,
            "volumeId": null,
            "linux": false,
            "volumes": [],
            "protected": false,
            "systemHold": false,
            "id": "512150d081814f93e4deb6055d66bcf4",
            "resourceType": "tagcategory",
            "storageProfiles": []
        }
    ]
}
```

Example 2: Get information about virtualize system instances with the storage view filter

Assume that you have a virtualized system VMware - Dallas1 which is vCenter Server. You get the {hypervisorHypervisorId} of it: 1001. The Python script snippet below will give you a list of instances:

```
hypervisor_hypervisor_id = "1001"    # VMware - Dallas

_params = {"from": "hlo"}

requests.get('https://' + spp_ipv4 + '/api/hypervisor/'
```

```
       + hypervisor_hypervisor_id + '/storagecontent',
       headers={...}, params=_params, data="", verify=...)
```

The response below with the HTTP status of 200 (OK) shows the list of instances under the virtualized system "VMware - Dallas1", with the storage view filter.


```
{
    "links": {...},
    "total": 1,
    "contents": [
        {
            "links": {...},
            "name": "PS_SE-Datacenter",
            "type": "datacenter",
            "association": null,
            "protectionInfo": null,
            "recoveryInfo": null,
            "recoveryPoint": null,
            "siteId": null,
            "autoProvisionedsubPolicyName": null,
            "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:vmware/hypervis
   or:1001/vdc:717dafb25bbbca30d8dc72ed3cbf81f0",
            "rbacPathId": 3440,
            "extInfo": null,
            "location": "/dallas.bluemachines.com.invalid",
            "metadataPath": "/MS:1001/DC:717dafb25bbbca30d8dc72ed3cbf81f0",
            "inHLO": true,
            "hypervisorType": "vmware",
            "hypervisorKey": "1001",
            "hypervisorManagementServerID": "1001",
            "nativeKey": "datacenter-2",
            "cloudType": null,
            "tags": [
                {
                    "name": "MGP Test2",
                    "key": "a3269bb10cfbe7a1588a5a3a5b3637de"
                }
            ],
            "tagCategories": [
                {
                    "name": "MGP-Test",
                    "key": "a6068ad43059b6b6fc3a0986ba7a0bed"
                }
            ],
            "metadataPathVmHost": null,
            "rbacPathVmHost": null,
            "rbacPathIdVmHost": null,
            "alternatePaths": [
                {
                    "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:vmware/
   hypervisor:1001/vtagcategory:a6068ad43059b6b6fc3a0986ba7a0bed/vtag:a3269bb10cfb
   e7a1588a5a3a5b3637de/vdc:717dafb25bbbca30d8dc72ed3cbf81f0",
                    "rbacPathId": 3535,
                    "metadataPath": "/MS:1001/TAGCATEGORY:a6068ad43059b6b6fc3a09
   86ba7a0bed/TAG:a3269bb10cfbe7a1588a5a3a5b3637de/DC:717dafb25bbbca30d8dc72ed3cbf
   81f0"
                }
            ],
            "accountName": null,
            "hypervisorHostKey": null,
            "nativeObject": {},
            "windows": false,
            "volumes": [],
            "linux": false,
            "volumeId": null,
            "protected": false,
            "systemHold": false,
            "id": "717dafb25bbbca30d8dc72ed3cbf81f0",
            "resourceType": "datacenter",
            "storageProfiles": []
        }
    ]
}
```

Example 3: Get information about virtualize system instances with the host view filter

Assume that you have a virtualized system VMware - Dallas1 which is vCenter Server. You get the {hypervisorHypervisorId} of it: 1001. The Python script snippet below will give you a list of instances:

```
hypervisor_hypervisor_id = "1001"    # VMware - Dallas

_params = {"from": "hlo"}

requests.get('https://' + spp_ipv4 + '/api/hypervisor/'
    + hypervisor_hypervisor_id + '/hostcontent'
    headers={...}, params=_params, data="", verify=...)
```

The response below with the HTTP status of 200 (OK) shows the list of all virtualized systems. As you see there are vCenter Servers ("type": "vmware") and Microsoft Hyper-V Servers ("hyperv") in the same list.


```
{
    "links": {...},
    "total": 1,
    "contents": [
        {
            "links": {...},
            "name": "PS_SE-Datacenter",
            "type": "datacenter",
            "association": null,
            "protectionInfo": null,
            "recoveryInfo": null,
            "recoveryPoint": null,
            "siteId": null,
            "autoProvisionedsubPolicyName": null,
            "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:vmware/hypervis
   or:1001/vdc:717dafb25bbbca30d8dc72ed3cbf81f0",
            "rbacPathId": 3440,
            "extInfo": null,
            "location": "/vmware-dallas1.bluemachines.com.invalid",
            "metadataPath": "/MS:1001/DC:717dafb25bbbca30d8dc72ed3cbf81f0",
            "inHLO": true,
            "hypervisorType": "vmware",
            "hypervisorKey": "1001",
            "hypervisorManagementServerID": "1001",
            "nativeKey": "datacenter-2",
            "cloudType": null,
            "tags": [
                {
                    "name": "MGP Test2",
                    "key": "a3269bb10cfbe7a1588a5a3a5b3637de"
                }
            ],
            "tagCategories": [
                {
                    "name": "MGP-Test",
                    "key": "a6068ad43059b6b6fc3a0986ba7a0bed"
                }
            ],
            "metadataPathVmHost": null,
            "rbacPathVmHost": null,
            "rbacPathIdVmHost": null,
            "alternatePaths": [
                {
                    "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:vmware/
   hypervisor:1001/vtagcategory:a6068ad43059b6b6fc3a0986ba7a0bed/vtag:a3269bb10cfb
   e7a1588a5a3a5b3637de/vdc:717dafb25bbbca30d8dc72ed3cbf81f0",
                    "rbacPathId": 3535,
                    "metadataPath": "/MS:1001/TAGCATEGORY:a6068ad43059b6b6fc3a09
   86ba7a0bed/TAG:a3269bb10cfbe7a1588a5a3a5b3637de/DC:717dafb25bbbca30d8dc72ed3cbf
   81f0"
                }
            ],
            "accountName": null,
            "hypervisorHostKey": null,
            "nativeObject": {},
            "windows": false,
```

```
                "volumes": [],
                "linux": false,
                "volumeId": null,
                "protected": false,
                "systemHold": false,
                "id": "717dafb25bbbca30d8dc72ed3cbf81f0",
                "resourceType": "datacenter",
                "storageProfiles": []
            }
        ]
    }
```

Adding VMWare vCenter Server or Hyper-V server

Register VMware vCenter Server or Hyper-V server with IBM Spectrum Protect Plus. A hypervisor inventory job will start automatically after a new vCenter Server is registered successfully.

For details of hypervisor inventory, follow the instructions in "Running an inventory job for virtualized systems" on page 64.

Method and URI

To register VMware vCenter Server or Hyper-V server with IBM Spectrum Protect Plus, use a POST method and a URI:

```
POST    https://{hostname|IP}/ngp/hypervisor
```

Parameters

None.

Data

Data 1: `hostAddress`

The hostname or the IPv4 address of VMware vCenter server or Hyper-V server.

- **Example value:** 10.0.0.10
- **Type:** String. Required. Available in the web user interface.

Data 2: `portNumber`

The port number to connect to the VMware vCenter Server or Hyper-V server

To connect to Hyper-V server without using the SSL, you must enable WinRM on the Hyper-V host system. For more information, see the IBM Spectrum Protect Plus User's and Installation Guide.

- **Example value:** You can use the following typical values:
- **Type:** Integer. Required. Available in the web user interface.

Data 3: `type`

The type of the virtualized system.

- **Value:** Use one of the following values:
- **Type:** System string. Required. Available in the web user interface.

Data 4: `username`

The `{identityUserHref}` of the existing user.

- **Example value:** https://10.0.0.100/api/identity/user/2139
- **Type:** String. Required. Available in the web user interface.

portNumber:

Usually, you will use either port: 443 (HTTPS with SSL) or 80 (HTTP without SSL). Ensure `sslConnection` is set to the correct value.

username:

You may use either an existing user (identity) or a new username and password.

An existing user can be defined by the identity URI of the corresponding user; and in this case, you do not have to enter the same password. The identity URI has the following syntax:

```
https://{hostname|IP}/api/identity/user/{identityUserId}
```

Here is an example:

```
https://10.0.0.100/api/identity/user/2101
```

opProperties:

Example: Register VMware vCenter server with IBM Spectrum Protect Plus

Assume that you want to register the following vCenter Server in IBM Spectrum Protect Plus:

- **Hostname/IP**: 10.0.0.10
- **Port**: 443 (HTTPS)
- **Username**: sarah.wiseman@bluemachines.com.invalid
- **Password**: #%'%<eyWqt'Z5EWg
- **Use SSL**: Yes
- **Maximum number of VM's to process concurrently per ESX server**: 3

A Python snippet that is similar to the following example can be used to register VMware vCenter server with IBM Spectrum Protect Plus:

```
_data = f'''{{
    "hostAddress":   "10.0.0.10",
    "portNumber":    443,
    "username":      "sarah.wiseman@bluemachines.com.invalid",
    "password":      "#%'%<eyWqt'Z5EWg",
    "sslConnection": true,
    "type":          "vmware",
    "opProperties":  {{
        "snapshotConcurrency":3
    }}
}}'''

requests.post('https://' + spp_ipv4 + '/ngp/hypervisor',
    headers={...}, params="", data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created).

```
{
    "statusCode": 201,
    "response": {
        "links": {...},
        "name": "10.0.0.10",
        "hostAddress": "10.0.0.10",
        "user": {
            "href": "https://10.0.0.100:-1/api/identity/user/2137"
        },
        "sslConnection": true,
        "portNumber": 443,
        "type": "vmware",
        "id": "1001",
        "uniqueId": "fa8f1a0f-9d01-44e6-b281-6c577cd920ae",
        "version": null,
        "build": null,
```

```
            "apiVersion": null,
            "properties": {},
            "logicalDelete": false,
            "accountName": null,
            "opProperties": {
                "snapshotConcurrency": 2,
                "veServerInfo": null
            },
            "rbacPath": "root:0/hypervisor.all:0/hypervisor.type:vmware/hypervisor:1
       001",
            "resourceType": "hypervisor"
        }
    }
```

If the connection with IBM Spectrum Protect Plus has failed, for example, when an incorrect host address or an unavailable port number was given, the request prompts a response that is structured as shown, with the HTTP status of 400 (Bad Request).

```
    {
        "statusCode": 400,
        "response": {
            "id": "HypervisorConnectionException",
            "description": "Could not connect to server 10.0.0.200: Exception caught
         trying to invoke method RetrieveServiceContent; nested exception is: \n\tjava.
       net.ConnectException: Connection refused (Connection refused)",
            "title": "Error"
        }
    }
```

If you enter an invalid username and password, the request prompts a response that is structured as shown, with the HTTP status of 401 (Unauthorized).

```
    {
        "statusCode": 401,
        "response": {
            "id": "HypervisorConnectionAuthException",
            "description": "Authentication failed. Ensure the username and password
       are correct. User must have all the vCenter VMware User privileges as document
       ed.",
            "title": "Error"
        }
    }
```

Unregistering a virtualized system

Unregister a virtualized system from IBM Spectrum Protect Plus.

Method and URI

To delete a hypervisor server with {hypervisorHypervisorId}, use a DELETE method and a URI:

```
    DELETE  https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}
```

Tip: To get a {hypervisorHypervisorId} based on the hostname or the IPv4 address, follow the information in “Getting a {hypervisorHypervisorId} ” on page 227.

Parameters

None.

Data

None.

Example: Unregister a vCenter Server

Assume that you have a virtualized system "10.0.0.10", which is one of the vCenter Servers, and you want to delete it. You get the {hypervisorHypervisorId}: 1003. Run the following command:

```
hypervisor_hypervisor_id = "1003"

requests.delete('https://' + spp_ipv4 + '/api/hypervisor/'
    + hypervisor_hypervisor_id,
    headers={...}, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 204 (No Content) and that you no longer see the VMwrae vCenter Server 10.0.010. The response body is empty.

Application Servers (File Systems, Containers, Cloud Management, Databases)

This chapter discusses common operations for all types of application servers and applications on them.

```
reference_application_inventory
reference_application_inventory_get
```

Running an inventory job for application server instances

IBM Spectrum Protect Plus supports application-level operations for the following solutions of application servers:

- File Systems
  - Microsoft Windows
- Containers
  - Kubernetes
  - OpenShift
- Cloud Management
  - Microsoft 365
- Databases
  - Db2
  - Exchange
  - MongoDB
  - Oracle
  - SQL

An inventory job for application servers will rebuild a list of application servers and application data on it such as databases that are registered in IBM Spectrum Protect Plus. An inventory job for application servers will start automatically as soon as you register a new application server on IBM Spectrum Protect Plus. In addition, every time you you make any changes in an application such as deploying new databases on it, you have to run an inventory job at IBM Spectrum Protect Plus to update its list

of application instances. To run an inventory job for virtualized systems (VMware, Hyper-V, and Amazon EC2), follow the instructions in “Running an inventory job for virtualized systems” on page 64.

Method and URI

To start an inventory job for application server instances, use a POST method and a URI, where `{jobId}` is the ID for the application server inventory job:

```
POST    https://{hostname|IPv4}/api/endeavour/job/{jobId}
```

Tip: To get the {jobId} for the application server inventory job, follow the information in “Getting a {jobId} ” on page 234, where the object name is `Application Server Inventory`. This {jobId} may vary by the IBM Spectrum Protect Plus appliance deployment.

Parameters

Parameter 1: `action`

Specify the type of action.

- **Value:** `start`
- **Type:** System string. Required. Available in the web interface.

Parameter 2: `actionname`

Specify the step of the action.

- **Value:** `start`
- **Type:** System string. Required. Available in the web interface.

Data

None.

Example: Start an inventory job for application server instances

Assume that you want to run an application server inventory job.

Create a function that converts the policy name, `Application Server Inventory`, into the job ID:

```
job_id = converter_job("Application Server Inventory")
print(job_id)
```

```
1004
```

A Python snippet that is similar to the following example can be used to start an inventory job for application server instances:

```
_params = {
    "action":     "inventory",
    "actionname": "start"
}

_response = requests.post('https://' + spp_ipv4 + '/api/endeavour/job/' + job_id
     headers={...}, params=_params, verify=...)
```

You will receive a response with no body and the HTTP status code of 204 (No Content).

Restriction: Do not determine the status of the application inventory jobs by referring to the HTTP status code in a response. This request only starts an inventory job and receives a response with the HTTP status

code of 204 (No Content) regardless of the status or the result. To see the status of the inventory jobs, follow the instructions in “Getting the status of an inventory job for application servers” on page 84.

Getting the status of an inventory job for application servers

You can get status information about a currently running inventory job and the previous inventory job for application servers.

Method and URI

To get information about current and previous application inventory jobs, use a GET method and a URI, where `{jobId}` is the ID for the application server inventory job:

```
GET     https://{hostname|IP}/api/endeavour/job/{jobId}
```

Parameters

None.

Data

None.

Example: Get status of an inventory job for application servers

Assume that the `{jobId}` of the application server inventory job is 1004.

A Python snippet that is similar to the following example can be used to request information about the present and previous application server inventory jobs:

```
job_id = "1004"  # Application Server Inventory

requests.get('https://' + spp_ipv4 + '/api/endeavour/job/' + job_id,
    headers={...}, verify=...)
```

Assume that you ran an application inventory job and send the request to see the status of it. Normally, the inventory job takes a few minutes to complete and so you should see the running status. In your browser, the blue **Run Inventory** button has been grayed out and reading **Inventory In Progress**. Here is an example response with the HTTP status code of 200 (OK).

```
{
    "links": {...},
    "name": "Application Server Inventory",
    "description": null,
    "policyId": "1003",
    "policyName": "Application Server Inventory",
    "type": "catalog",
    "typeDisplayName": "Catalog",
    "subType": "application",
    "subTypeDisplayName": "Application",
    "serviceId": "serviceprovider.catalog.application",
    "displayName": "Application Server Inventory",
    "status": "RUNNING",
    "statusDisplayName": "Running",
    "lastSessionStatus": "PARTIAL",
    "lastSessionStatusDisplayName": "Partial",
    "triggerIds": [
        "1003"
    ],
    "triggerData": [
        {
            "triggerId": "1003",
            "triggerInfo": null
        }
    ],
    "lastRunTime": 1578684906215,
```

```
    "nextFireTime": 1578729600000,
    "lastSessionDuration": 20549,
    "tenantId": 1000,
    "actions": null,
    "statistics": null,
    "lastrun": {
        "sessionId": "1578684905188",
        "jobName": "Application Server Inventory",
        "type": "catalog",
        "subType": "application",
        "serviceId": "serviceprovider.catalog.application",
        "start": 1578684906215,
        "end": 1578684926764,
        "duration": 20549,
        "status": "PARTIAL",
        "results": "Started",
        "properties": null,
        "numTasks": 2,
        "previousLastTask": 0,
        "lastUpdate": 1578684926769,
        "percent": 0,
        "policySnapshot": null
    },
    "id": "1003"
}
```


In the example response above, the `status` value reads `RUNNING` which means the application inventory job was running at the time you sent the request. You can also see the information about the previous hypervisor inventory job.

A few minutes later, you send the same request again to see the status. At this time, the button in the browser comes back to the original, blue **Run Inventory** button. Here is an example response with the HTTP status code 200 (OK). As you see, the `status` value reads `IDLE`.


```
{
    "links": {...},
    ...
    "status": "IDLE",
    "statusDisplayName": "Idle",
    "lastSessionStatus": "PARTIAL",
    "lastSessionStatusDisplayName": "Partial",
    "triggerIds": [
        "1004"
    ],
    "triggerData": [
        {
            "triggerId": "1004",
            "triggerInfo": null
        }
    ],
    "lastRunTime": 1578681060744,
    "nextFireTime": 1578729600000,
    "lastSessionDuration": 529178,
    "tenantId": 1000,
    "actions": null,
    "statistics": null,
    "lastrun": {...},
    ...
}
```


Restriction: Do not determine the status of the application inventory jobs by referring to the HTTP status code in a response. This request will make a response with the HTTP status code of 200 (OK) regardless of the status. Instead, refer to the relevant key-value pairs in the response to check the status.

Adding application servers and hosts

Register application servers and hosts with IBM Spectrum Protect Plus.

When you register an application server or host, you must provide information to verify the identity of the server. Application servers are Microsoft Windows-based, such as Exchange, File System, and SQL, or are Linux-based, such as Db2, Microsoft 365, MongoDB or Oracle. Virtual machines hosts on hypervisors may also be Windows-based or Linux-based operating systems. The information that is used to verify

the identity of the server is the SSL certificate thumbprint for Windows and the SSH key, SSH MD5 key fingerprint, and SSH key type for Linux.

This information can be obtained from the application server or host directly or by using the IBM Spectrum Protect Plus REST API. After the information is obtained, it is used to add the application server or host to IBM Spectrum Protect Plus.

The following table details what is required for Windows and Linux application servers and hosts:

Windows-based application servers	Linux-based application servers
SSL certificate thumbprint	SSH key
	SSH MD5 key fingerprint
	SSH key type

Method and URI

To register a Windows-based or Linux-based server or host with IBM Spectrum Protect Plus, use a POST method and a URI:

```
POST    https://{hostname|IP}/api/appserver
```

Parameters

Parameter: `action`

You must specify the action parameter to obtain the SSL certificate thumbprint for Windows-based application servers or hosts and to obtain the SSH key, SSH MD5 key fingerprint, and SSH key type for Linux-based application servers or hosts.

- **Value:** `getserverkey`
- **Type:** String. Required. Available in the web user interface.

Data

Data 1: `hostAddress`

The hostname or the IPv4 address of application server.

- **Example value:** 10.0.0.10
- **Type:** String. Required. Available in the web user interface.

Data 2: `osType`

The operating system (OS) type of the application server or host being registered.

- **Example value:** linux
- **Type:** String. Required. Available in the web user interface.

  You can use one of the following values:

  – linux
  – windows

Data 3: `osProperties`

For Windows-based application servers and hosts, the SSL certificate thumbprint must be supplied in the POST body as the `keyFingerprint`. For Linux-based application servers and hosts, the SSH key, SSH MD5 key fingerprint, and SSH key type must be supplied in the POST body as the `keyType`, `key`, and `keyFingeprint` respectively.

Tip: To use SSL certificates with the Windows-based server or host as described in this topic, you must set the IBM Spectrum Protect Plus system to use port 5986 for the **Windows Clients Port (WinRM) used for application and file indexing** option in Global Preferences and enable the HTTPS transport mode on ts host. For more information, see the IBM Spectrum Protect Plus User's and Installation Guide. If port 5985 is defined, which is the default, you do not have to supply the SSL certificate thumbprint when adding a Windows-based host.

The following must be provided for adding an application server or host:

- `keyType`
  - **Example value:** `ssh-rsa`
  - **Type:** String. Required for Linux-based application servers and hosts only.
- `key`
  - **Example value:** `AABBB3NzaC1yc2EAAAADAQABBBABAQCsaG+6Qj+dAsNOOU0o5I2GzYjpFT573184lJAfKbUK2fF47IRnheH9lgDd4wOK0Jto81YTKqjWvSioQGOsDcR+6O2b5QiKl6H0OylTu06CTNn66whMhifAoq/k6qOdk6PtxJR6FoSiG9RhF6QicmngSgBWNn2QujJnmWdui5b1VbNxmPsqwteYSp85feLpr0rvezQ7Ekna6mB1CgjL3xwErMw+cZBmgfuip9ZzEI3i0/FPwJzAl5pdF6k56b8dort9RajgnmJOoXpTjUJjt8R2g6LTNKIW8QCn4UpL660l9b4tetixFLTt92ehRy1PJigQC9C+maa3ShIgXN1KWK+l`
  - **Type:** String. Required for Linux-based application servers and hosts only.
- `keyFingerprint`
  - **Example value:** `07CF2B1B1E3A755AA6F5EFABD99E457A17E20A35` (Windows) or `55:c4:03:f0:b8:e6:0a:fe:44:e5:7f:e8:e6:52:09:3e` (Linux)
  - **Type:** String. Required.

Example 1: Getting a certificate thumbprint for a Windows-based application server or host

Prior to registering a Windows-based application server or host, you must obtain the SSL certificate thumbprint.

In this example, the Windows-based SSL certificate thumbprint is obtained:

- **Hostname/IP**: 10.0.0.10
- **OS Type**: windows

A Python snippet that is similar to the following example can be used to obtain the Windows-based server or host SSL certificate thumbprint:

```
_data = f'''{
    "hostAddress":"10.0.0.10",
    "osType":"windows"
}}'''

requests.post('https://' + spp_ipv4 + '/api/appserver?action=getserverkey',
    headers={...}, params="", data=_data, verify=...)
```

The request prompts a response with the HTTP status of 200 (OK).

```
{
 "keyFingerprint": "07CF2B1B1E3A755AA6F5EFABD99E457A17E20A35"
}
```

Example 2: Add a Windows-based server or host in IBM Spectrum Protect Plus

Assume that you want to register the following Windows-based server or host in IBM Spectrum Protect Plus. In this example, a File System server will be registered:

- **Hostname/IP**: 10.0.0.10
- **OS Type**: windows

A Python snippet that is similar to the following example can be used to register a Windows-based server or host with IBM Spectrum Protect Plus:

```
_data = f'''{
    "addToCatJob": true,
    "hostAddress": "10.0.0.10",
    "application": true,
    "script": false,
    "osuser": {
        "href": "https://10.0.0.12/api/identity/user/2193"
    },
    "osType": "windows",
    "applicationType": "file",
    "name": "10.0.0.10",
    "opProperties": {
        "maxConcurrency": 10,
        "keyFingerprint": "07CF2B1B1E3A755AA6F5EFABD99E457A17E20A35"
}}'''

requests.post('https://' + spp_ipv4 + '/api/appserver',
    headers={...}, params="", data=_data, verify=...)
```

The request prompts a response with the HTTP status of 201 (Created).

Example 3: Getting a server key for a Linux-based application server or host

Prior to registering a Linux-based application server or host, you must obtain the SSH key, SSH MD5 key fingerprint, and SSH key type.

In this example, the Linux-based SSH key, SSH MD5 key fingerprint, and SSH key type is obtained:

- **Hostname/IP**: 10.0.0.10
- **OS Type**: linux

A Python snippet that is similar to the following example can be used to obtain the Linux-based server or host SSH key, SSH MD5 key fingerprint, and SSH key type:

```
_data = f'''{
    "hostAddress":"10.0.0.10",
    "osType":"linux"
}}'''

requests.post('https://' + spp_ipv4 + '/api/appserver?action=getserverkey',
    headers={...}, params="", data=_data, verify=...)
```

The request prompts a response with the HTTP status of 200 (OK).

```
{
    "key":
"AABBB3NzaC1yc2EAAAADAQABBBABAQCsaG+6Qj+dAsNOOU0o5I2GzYjpFT573184lJAfKbUK2fF47IRnheH9lgDd4wOK0Jt
o81YTKqjWvSioQGOsDcR+6O2b5QiKl6H0OylTu06CTNn66whMhifAoq/
k6qOdk6PtxJR6FoSiG9RhF6QicmngSgBWNn2QujJnmWdui5b1VbNxmPsqwteYSp85feLpr0rvezQ7Ekna6mB1CgjL3xwErMw
+cZBmgfuip9ZzEI3i0/
FPwJzAl5pdF6k56b8dort9RajgnmJOoXpTjUJjt8R2g6LTNKIW8QCn4UpL660l9b4tetixFLTt92ehRy1PJigQC9C+maa3Sh
IgXN1KWK+l",
    "keyFingerprint": "55:c4:03:f0:b8:e6:0a:fe:44:e5:7f:e8:e6:52:09:3e",
    "keyType": "ssh-rsa"
}
```

Example 4: Register a Linux-based server or host in IBM Spectrum Protect Plus

Assume that you want to register the following Linux-based server or host in IBM Spectrum Protect Plus. In this example, an Oracle server will be registered:

- **Hostname/IP**: 10.0.0.10
- **OS Type**: linux

A Python snippet that is similar to the following example can be used to register a Linux-based server or host with IBM Spectrum Protect Plus:

```
   _data = f'''{
        "addToCatJob": true,
        "hostAddress": "10.0.0.10",
        "application": true,
        "script": false,
        "osuser": {
             "href": "https://10.0.0.12/api/identity/user/2185"
        },
        "osType": "linux",
        "applicationType": "oracle",
        "name": "10.0.0.10",
        "opProperties": {
             "maxConcurrency": 10,
             "keyType": "ssh-rsa",
             "key":
"AABBB3NzaC1yc2EAAAADAQABBBABAQCsaG+6Qj+dAsNOOU0o5I2GzYjpFT573184lJAfKbUK2fF47IRnheH9lgDd4wOK0Jt
o81YTKqjWvSioQGOsDcR+6O2b5QiKl6H0OylTu06CTNn66whMhifAoq/
k6qOdk6PtxJR6FoSiG9RhF6QicmngSgBWNn2QujJnmWdui5b1VbNxmPsqwteYSp85feLpr0rvezQ7Ekna6mB1CgjL3xwErMw
+cZBmgfuip9ZzEI3i0/
FPwJzAl5pdF6k56b8dort9RajgnmJOoXpTjUJjt8R2g6LTNKIW8QCn4UpL660l9b4tetixFLTt92ehRy1PJigQC9C+maa3Sh
IgXN1KWK+l",
             "keyFingerprint": "55:c4:03:f0:b8:e6:0a:fe:44:e5:7f:e8:e6:52:09:3e"
   }}'''

   requests.post('https://' + spp_ipv4 + '/api/appserver',
       headers={...}, params="", data=_data, verify=...)
```

The request prompts a response with the HTTP status of 201 (Created).

Databases (Db2, Exchange, MongoDB, Oracle, SQL)

IBM Spectrum Protect Plus supports backup and restore for five types of database solutions: IBM Db2, Microsoft Exchange Server, MongoDB, Oracle Database and Microsoft SQL Server.

```
   refmanageprotection_databases_get
```

Getting information for databases

You can get a list of application servers that host databases. You can also specify one application server and obtain information about the specified application server.

Method and URI

To get information about all application instances of a specific application, use a GET method and a URI:

```
   GET     https://{hostname|IP}/api/application/{applicationName}/instance
```

To get information about a specific application instance, use a GET method and a URI:

```
   GET     https://{hostname|IP}/api/application/{applicationName}
      /instance/{applicationInstanceId}
```

To get information about databases of a certain application instance, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/application/{applicationName}
   /instance/{applicationInstanceId}/applicationview
```

To get information about a specific database of a certain application instance, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/application/{applicationName}
   /instance/{applicationInstanceId}
   /database/{applicationDatabaseId}
```

Tip: To get an {applicationName} value based on the type of the application, follow the information in "Getting an {applicationName}" on page 222. To get an {applicationInstanceId} value based on the name of the application instances, see the information in "Getting an {applicationInstanceId}" on page 223.

Parameters

Parameter 1: `from`

- **Value:** `hlo`
- **Type:** System string. Required.

Example 1: Get information about application instances

Assume that you want to see a list of Db2 application servers that are registered with IBM Spectrum Protect Plus.

A Python snippet that is similar to the following example can be used to request information about all Db2 application instances:

```
application_name = "db2"

requests.get('https://' + spp_ipv4 + '/api/application/' + application_name
    + '/instance'
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK). You can get information about application instances of other types of applications in a similar way.

```
{
    "links": {...},
    "total": 2,
    "instances": [
        {
            "links": {...},
            "id": "f794d5b0234ffa5271738a4bd9fbd905",
            "type": "db2",
            "osType": "linux",
            "isOwnerNode": false,
            "host": "10.7.7.1",
            "name": "db2dallas1.bluemachines.com / db2americas",
            "applicationFullName": "IBM Db2",
            "version": "11.1.3",
            "providerNodeId": "1001",
            "sessionId": 1571343979215,
            "jobId": "1005",
            "resourceType": "applicationinstance",
            "rbacPath": "root:0/app:0/app.type:db2/appinst:f794d5b0234ffa5271738a
   4bd9fbd905",
            "appServerVmInfos": [
                {
                    "appServerId": "1001",
                    "appServerName": "10.7.7.1",
                    "vmId": null,
                    "vmName": null,
                    "vmLocation": null,
```

```
                    "hypervisorServerId": null,
                    "hypervisorServerName": null,
                    "hypervisorHost": null
                }
            ],
            "capabilities": {
                "resolveLogBackupLocations": false,
                "presnapshot": null,
                "snapshot": null,
                "postsnapshot": null,
                "enablelogbackup": null,
                "disablelogbackup": null,
                "logtruncate": null,
                "recordbackup": null,
                "masking": null,
                "prerestore": null,
                "restore": null,
                "postrestore": null,
                "restorecleanup": null
            },
            "location": "/db2dallas1.bluemachines.com / db2americas",
            "metadataPath": "/DBINST:f794d5b0234ffa5271738a4bd9fbd905",
            "storageProfiles": []
        },
        {
            "links": {...},
            "id": "26e734b8e0ac6e3afa52ba1cd2bd6e93",
            "type": "db2",
            "osType": "aix",
            "isOwnerNode": false,
            "host": "10.0.7.2",
            "name": "db2singapore1.bluemachines.com / db2apac",
            ...
        }
    ]
}
```

Example 2: Get information about databases

Assume that you want to get information about all databases in the following Db2 application instance:

- **{applicationInstanceName}:** db2dallas1.bluemachines.com / db2americas.
- **{applicationInstanceId}:** f794d5b0234ffa5271738a4bd9fbd905

A Python snippet that is similar to the following example can be used to request information about all databases in the application instance:

```
application_name = "db2"
application_instance_id = "f794d5b0234ffa5271738a4bd9fbd905"

requests.get('https://' + spp_ipv4 + '/api/application/' + application_name
    + '/instance/' + application_instance_id + '/applicationview/'
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK). You can get information about application instances of other types of applications in a similar way.

```
{
    "links": {...,}
    "total": 2,
    "contents": [
        {
            "links": {...},
            "name": "DB_SALES",
            "type": "db2",
            "resourceType": "database",
            "id": "b91891e551ee0ac0765472ac0e3ba041",
            "host": "10.7.7.1",
            "providerNodeId": "1001",
            "sessionId": 1571343979215,
            "jobId": "1005",
            "rbacPath": "root:0/app:0/app.type:db2/appinst:f794d5b0234ffa5271738
   a4bd9fbd905/appdb:b91891e551ee0ac0765472ac0e3ba041",
            "usedSize": 0,
```

```
            "totalSize": 0,
            "dataVolumeIds": [],
            "logVolumeIds": [],
            "eligibility": {
                "backup": {
                    "eligible": true,
                    "reason": ""
                },
                "logbackup": {
                    "eligible": true,
                    "reason": ""
                }
            },
            "metadataPath": "/DBINST:f794d5b0234ffa5271738a4bd9fbd905/DB:b91891e
   551ee0ac0765472ac0e3ba041",
            "storageProfiles": [],
            "paths": [],
            "location": "/db2dallas1.bluemachines.com / db2americas/DB_SALES"
        },
        {
            "links": {...},
            "name": "DB_SERVICES",
            "type": "db2",
            "resourceType": "database",
            "id": "8c52d9437edfe68896ef9d1d91f77bfd",
            ...
        }
    ]
}
```

Example 3: Get information about a specific database

Assume that you want to get information about the following database:

- **{applicationInstanceName}:** db2dallas1.bluemachines.com / db2americas.
- **{applicationInstanceId}:** f794d5b0234ffa5271738a4bd9fbd905
- **Database name:** DB_SALES

A Python snippet that is similar to the following example can be used to request information about all databases in the application instance:

```
application_name = "db2"
application_instance_id = "f794d5b0234ffa5271738a4bd9fbd905"

requests.get('https://' + spp_ipv4 + '/api/application/' + application_name
    + '/instance/' + application_instance_id + '/applicationview/'
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK). You can get information about application instances of other types of applications in a similar way.

```
{
    "links": {...,}
    "total": 2,
    "contents": [
        {
            "links": {...},
            "name": "DB_SALES",
            "type": "db2",
            "resourceType": "database",
            "id": "b91891e551ee0ac0765472ac0e3ba041",
            "host": "10.7.7.1",
            "providerNodeId": "1001",
            "sessionId": 1571343979215,
            "jobId": "1005",
            "rbacPath": "root:0/app:0/app.type:db2/appinst:f794d5b0234ffa5271738
   a4bd9fbd905/appdb:b91891e551ee0ac0765472ac0e3ba041",
            "usedSize": 0,
            "totalSize": 0,
            "dataVolumeIds": [],
            "logVolumeIds": [],
            "eligibility": {
                "backup": {
```

```
                    "eligible": true,
                    "reason": ""
                },
                "logbackup": {
                    "eligible": true,
                    "reason": ""
                }
            },
            "metadataPath": "/DBINST:f794d5b0234ffa5271738a4bd9fbd905/DB:b91891e
    551ee0ac0765472ac0e3ba041",
            "storageProfiles": [],
            "paths": [],
            "location": "/db2dallas1.bluemachines.com / db2americas/DB_SALES"
        },
        {
            "links": {...},
            "name": "DB_SERVICES",
            "type": "db2",
            "resourceType": "database",
            "id": "8c52d9437edfe68896ef9d1d91f77bfd",
            ...
        }
    ]
}
```

Databases: SQL

You can complete operations that are specifically available for SQL Server instances with the REST API, as described in the following sections.

Getting information about log backups for SQL Servers

You can get information about log backups for SQL Servers. This action is available only in the REST API.

Method and URI

To get information about log backups for SQL Servers, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/application/sql/logbackupstatus
```

Parameters

None.

Data

None.

Example: Get information about log backups for SQL Servers

Assume that you registered SQL Servers with IBM Spectrum Protect Plus and ran log backup jobs.

A Python snippet that is similar to the following example can be used to request information about the log backup status for each database:

```
requests.get('https://' + spp_ipv4 + '/api/application/sql/logbackupstatus',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "total": 4,
    "results": [{
            "links": {...},
            "databaseName": "Sales_Americas",
            "databasePk": "28b50719ebbcf817b927ac42df35e334",
```

```
                "instanceName": "SQL-DALLAS1",
                "instancePk": "3bb74f48d",
                "status": "completed",
                "appType": "sql",
                "pk": "9141b6b3ccbee029b5817eff7190b1b1",
                "name": "SQL-DALLAS1Sales_Americas",
                "time": 1563289224690,
                "appMetadata": {
                    "startTransactionId": "32000000021200001",
                    "endTransactionId": "32000000025900001"
                },
                {
                ...,
                "databaseName": "Support_Americas",
                ...,
                "instanceName": "SQL-DALLAS1",
                ...
                },
                {
                ...,
                "databaseName": "Sales_Apac",
                ...,
                "instanceName": "SQL-SINGAPORE1",
                ...
                },
                {...},
            }
        ]
    }
```

Starting an ad hoc log backup job for SQL Servers

You can start an ad hoc job for SQL Server's log backup:

Method and URI

To start an ad hoc log backup job for SQL Server objects, use a POST method and a URI for one of the following types of targets:

- All databases on an SQL Server instance that does not use the Always On availability groups (AG) feature:

```
POST    https://{hostname|IPv4}/api/application/sql/
   /instance/{applicationInstanceId}
```

- A specific database on a standalone instance or a failover cluster instance (FCI):

```
POST    https://{hostname|IPv4}/api/application/sql/
   /instance/{applicationInstanceId}
   /database/{applicationDatabaseId}
```

- All databases in an AG:

```
POST    https://{hostname|IPv4}/api/application/sql/
   /databasegroup/{applicationDatabasegroupId}
```

- A specific database in an AG:

```
POST    https://{hostname|IPv4}/api/application/sql/
   /databasegroup/{applicationDatabasegroupId}
   /database/{applicationDatabaseId}
```

Parameters

Parameter 1: `action`

Start an ad hoc log backup job.

- **Value:** backuplogs
- **Type:** System string. Required.

Data

None.

Example: Start an ad hoc log backup job for SQL Servers

Assume that you registered SQL Servers with IBM Spectrum Protect Plus and you want to start an ad hoc job for log backup.

A Python snippet that is similar to the following example can be used to start an ad hoc job for log backup:

```
_params = {"action": "backuplogs"}

requests.post('https://' + spp_ipv4 + '/api/application/sql/logbackupstatus',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).


```
{
  "issuccess": true,
  "partial": false,
  "message": null,
  "instances": [
    {
      "name": "SQL-DALLAS1",
      "primaryKey": "3901715654",
      "databases": [
        {
          "name": "Sales_Americas",
          "primaryKey": "bdf61739552dc12302a4a15bb49a7bf0",
          "issuccess": true,
          "message": null,
          "recoveryTimestamp": null,
          "recoveryTransactionId": null,
          "appMetadata": null,
          "logMetadata": null,
          "transferDuration": 0,
          "transferSize": 0,
          "backupPaths": null
        },
        {
          "name": "Support_Americas",
          "primaryKey": "95e6eff67da3d3813c86675ba978b538",
          "issuccess": true,
          "message": null,
          "recoveryTimestamp": null,
          "recoveryTransactionId": null,
          "appMetadata": null,
          "logMetadata": null,
          "transferDuration": 0,
          "transferSize": 0,
          "backupPaths": null
        }
      ]
    },
    {
      "name": "SQL-DALLAS1",
      "primaryKey": "3901715654",
      "databases": [
        {
          "name": "Sales_Americas",
          "primaryKey": "bdf61739552dc12302a4a15bb49a7bf0",
          "issuccess": true,
          "message": null,
          "recoveryTimestamp": null,
          "recoveryTransactionId": null,
          "appMetadata": null,
          "logMetadata": null,
          "transferDuration": 0,
          "transferSize": 0,
          "backupPaths": null
```

```
          },
          {
            "name": "Support_Americas",
            "primaryKey": "95e6eff67da3d3813c86675ba978b538",
            "issuccess": true,
            "message": null,
            "recoveryTimestamp": null,
            "recoveryTransactionId": null,
            "appMetadata": null,
            "logMetadata": null,
            "transferDuration": 0,
            "transferSize": 0,
            "backupPaths": null
          }
        ]
      }
    ]
  }
```

Chapter 5. System Configuration

Backup Storage: Disk

IBM Spectrum Protect Plus supports one type of disk storage for backup: IBM vSnap servers.

Getting disk storage information about vSnap servers

You can get a list of all vSnap servers and information about each vSnap server. You can get vSnap server information from the cache data that is stored in the IBM Spectrum Protect Plus environment. Or, you can prompt IBM Spectrum Protect Plus to collect the latest information from a specific vSnap server.

Notice: Beginning with Version 10.1.7, the IBM Spectrum Protect Plus virtual appliance does not have an onboard vSnap server. You might see "localhost" in a list of vSnap servers if you upgraded IBM Spectrum Protect Plus from Version 10.1.6 or earlier. In this case, migrate your data from the onboard vSnap server, which is designated as "localhost", to external vSnap servers, object storage (Amazon S3, Microsoft Azure Blob Object Storage, IBM Cloud Object Storage, or S3-compatible storage), or replication servers (IBM Spectrum Protect) and delete the onboard vSnap server. This action is required because an onboard vSnap server must not be used in a production environment.

Method and URI

To get information about all vSnap servers using cached records in IBM Spectrum Protect Plus, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/storage
```

To get information about a specific vSnap server using cached records in IBM Spectrum Protect Plus, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/storage/{storageId}
```

Tip: To get a {storageId} value based on its name, follow the information in “Getting a {storageId}” on page 242.

To request the latest information about a specific vSnap server, use a POST method and a URI:

```
POST    https://{hostname|IP}/api/storage/{storageId}
```

Restriction: With this POST method request, you cannot retrieve information about all vSnap servers at once. Typically, the request takes longer than the GET requests that collect vSnap server information from IBM Spectrum Protect Plus.

Parameters

There is no parameters for the GET requests to retrieve vSnap information from IBM Spectrum Protect Plus.

For the POST request to retrieve information from a vSnap server, use the following parameter:

Parameter 1: `action`

Start an action to retrieve vSnap server information from the vSnap servers.

- **Value:** `refresh`
- **Type:** System string. Required. Available in the web user interface.

Data

None.

Example 1: Get information about all vSnap servers

The following Python snippet requests a list of all vSnap servers and their information: As mentioned, the information here might be old and inaccurate.

```
requests.get('https://' + spp_ipv4 + '/api/storage',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).


```
{
    "links": {...},
    "total": 2,
    "storages": [
        {
            "links": {...},
            "resourceType": "storage",
            "type": "vsnap",
            "typeDisplayName": "vSnap",
            "site": "1000",
            "name": "10.0.1.1",
            "storageId": "2101",
            "user": {
                "href": "https://10.0.0.100:-1/api/identity/user/2120"
            },
            "rbacPath": "root:0/site:0/site:1000/site.all.storage:1000/storage:2
   101",
            "hostAddress": "10.0.1.1",
            "portNumber": 8900,
            "sslConnection": true,
            "initializeStatus": "Ready",
            "initializeStatusDisplayName": "Ready",
            "storageProfiles": null,
            "version": "10.1.8-1900",
            "capacity": {
                "free": 0,
                "total": 0,
                "updateTime": 1576117254291
            },
            "activeDirectoryInfo": null,
            "demo": false,
            "maxStreams": null,
            "isReady": true,
            "id": "2101"
        },
        {
            "links": {...},
            "resourceType": "storage",
            "type": "vsnap",
            "typeDisplayName": "vSnap",
            "site": "1001",
            "name": "10.0.1.2",
            "storageId": "2102",
            "user": {
                "href": "https://10.0.0.100:-1/api/identity/user/2120"
            },
            "rbacPath": "root:0/site:0/site:1001/site.all.storage:1000/storage:2
   101",
            "hostAddress": "10.0.1.2",
            "portNumber": 8900,
            "sslConnection": true,
            "initializeStatus": "Ready",
            "initializeStatusDisplayName": "Ready",
            "storageProfiles": null,
            "version": "10.1.8-1900",
            "capacity": {
                "free": 0,
                "total": 0,
                "updateTime": 1576117254291
            },
            "activeDirectoryInfo": null,
```

```
            "demo": false,
            "maxStreams": null,
            "isReady": true,
            "id": "2102"
        }
    ]
}
```

Example 2: Get the latest information about a specific vSnap server

Assume that you added a vSnap server, vsnap-dallas1 ({storage_id} 2001) to IBM Spectrum Protect Plus. The following Python snippet requests information of the vSnap server. This POST method request retrieves the latest vSnap server information from the target vSnap server:

```
storage_id = "2101"
_params = {"action": "refresh"}

requests.post('https://' + spp_ipv4 + '/api/storage/' + storage_id,
    headers={...}, params=_params, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created).

```
{
    "links": {...},
    "resourceType": "storage",
    "type": "vsnap",
    "typeDisplayName": "vSnap",
    "site": "1000",
    "name": "10.0.1.1",
    "storageId": "2101",
    ...,
    "id": "2101",
}
```

Adding disk storage: vSnap server

Add a vSnap server to IBM Spectrum Protect Plus.

Method and URI

To add a vSnap server to IBM Spectrum Protect Plus, use a POST method and a URI:

```
POST    https://{hostname|IPv4}/ngp/storage
```

Parameters

None.

Data

Data 1: `hostAddress`

The hostname or the IPv4 address of the vSnap server.

- **Example value:** 10.0.1.1
- **Type:** String. Required. Available in the web user interface.

Data 2: `siteId`

The {siteId} of the site to associate with the vSnap server.

- **Example value:** 1000
- **Type:** String. Required. Available in the web user interface.

Tip: To get a {siteId} value based on its name, follow the instructions in "Getting a {siteId}" on page 240.

Data 3: `type`

The type of the disk storage. Use the vSnap server which is only available disk storage for IBM Spectrum Protect Plus.

- **Value:** `vsnap`
- **Type:** System string. Required.

Data 4: `username`

The username of the vSnap server.

- **Example value:** sarah-vsnap1
- **Type:** String. Required. Available in the web user interface.

Data 5: `password`

The password of the vSnap server.

- **Example value:** 6bxPYs6eQNa!V85R
- **Type:** String. Required. Available in the web user interface.

Data 6: `portNumber`

The port number to connect to the vSnap server.

- **Example value:** 8900
- **Type:** Integer. Required.

Data 7: `sslConnection`

Indication of whether SSL is used to connect with the vSnap server.

- **Example value:** true
- **Type:** Boolean. Required.

Example: Add a vSnap server

Assume that you want to add the following vSnap server to IBM Spectrum Protect Plus:

- **Hostname/IP:** 10.0.1.1
- **Site:** Primary (`{siteId}` 1000)
- **User name:** sarah-vsnap1
- **Password:** 6bxPYs6eQNa!V85R
- **Port:** 8900
- **SSL connection:** Enable

A Python snippet that is similar to the following example can be used to add the vSnap server to IBM Spectrum Protect Plus.

```
_data = f'''{{
     "hostAddress":   "10.0.1.1",
     "siteId":        "1000",
     "type":          "vsnap",
     "username":      "sarah-vsnap1",
     "password":      "6bxPYs6eQNa!V85R",
     "portNumber":    8900,
     "sslConnection": true
}}'''

requests.post('https://' + spp_ipv4 + '/ngp/storage',
    headers={...}, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created). Review the response to ensure that the vSnap server was added.


```
{
    "statusCode": 201,
    "response": {
        "links": {...},
        "resourceType": "storage",
        "type": "vsnap",
        "typeDisplayName": "vSnap",
        "site": "1000",
        "name": "10.0.1.1",
        "storageId": "2101",
        "user": {
            "href": "https://10.0.1.1:-1/api/identity/user/2120"
        },
        "rbacPath": "root:0/site:0/site:1000/site.all.storage:1000/storage:2101"
   ,
        "hostAddress": "10.0.1.1",
        "portNumber": 8900,
        "sslConnection": true,
        "initializeStatus": "Not Initialized",
        "initializeStatusDisplayName": "Not Initialized",
        "storageProfiles": null,
        "version": "10.1.8-1824",
        "capacity": null,
        "activeDirectoryInfo": null,
        "demo": false,
        "maxStreams": null,
        "isReady": false
        "statistics": {
            "storageId": "2101",
            "sizeTotal": 0,
            "sizeUsed": 0,
            "sizeFree": 0,
            "writable": true,
            "site": "1000",
            "description": null,
            "compressionRatio": "0.00",
            "deduplicationRatio": "0.00",
            "type": "vsnap",
            "errorDescription": null,
            "time": null,
            "lastUpdateTimestamp": null,
            "pools": []
        }
    }
}
```


Initializing a vSnap server

Initialize a new vSnap server you have added in three methods: (1) using the REST API, (2) using the browser interface, and (3) using the shell session on the vSnap server. This section describes the first method. For the second and the third methods, see the IBM Spectrum Protect Plus User's and Installation Guide. You can enable or disable encryption when you initialize the vSnap server.

Method and URI

To initialize the vSnap server with or without encryption, use a POST method and a URI:

```
POST    https://{hostname|IP}/api/storage/{storageId}/management
```

Tip: To get a {storageId} value based on its name, follow the information in “Getting a {storageId}” on page 242.

Parameters

Parameter 1: `action`

Start an action to initialize the vSnap server with encryption.

- **Value:** Use an either one of the following values:

Value	Description
initwithencryption	Initialize the vSnap server with encryption
init	Initialize the vSnap server without encryption

- **Type:** System string. Required. Available in the web user interface.

Data

Data 1: `async`

- **Value:** true
- **Type:** Boolean. Required.

Example: Initialize a vSnap server with encryption

Assume that you added a vSnap server, vsnap-dallas1 (`{storage_id}` 2001) to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to initialize the vSnap server with encryption:

```
storage_id = "2101"

_params = {"action": "initwithencryption"}

_data = f'''{{
    "async": true
}}'''

requests.post('https://' + spp_ipv4 + '/api/storage/' + storage_id
    + '/management',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "resourceType": "server",
    "fqdn": "vsnap-centos.dallas.bluemachines.com.invalid",
    "hostname": "vsnap-dallas1",
    "id": "2101",
    "init_status": "Initializing",
    "nfs_version": "1.3.0-0.61.el7",
    "nginx_version": "1.12.2-3.el7",
    "os_name": "CentOS Linux",
    "os_version": "7.8.2003",
    "samba_version": "4.10.4-11.el7_8",
    "uwsgi_version": "2.0.17.1-2.el7",
    "vsnap_version": "10.1.8-2000",
    "zfs_version": "Not Installed",
    "providerNodeId": "2101"
}
```

Rescanning disk storage: vSnap server

If IBM Spectrum Protect Plus reports that a vSnap server is reaching its storage capacity, the vSnap storage pool must be expanded. To expand a vSnap storage pool, you must first add virtual or physical disks on the vSnap server, either by adding virtual disks to the vSnap virtual machine or adding physical disks to the vSnap physical server. See the vSphere documentation for information about creating additional virtual disks.

Method and URI

To rescan a vSnap server with {storageId}, use a POST method and a URI:

```
POST    https://{hostname|IPv4}/api/storage/{storageId}
```

To get information about vSnap server disks that are not registered with IBM Spectrum Protect Plus, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/storage/{storageId}/management/disk
```

Parameters

Parameter 1: action

Rescan the vSnap server.

- **Value:** rescan
- **Type:** System string. Required. Available in the web user interface.

Data

None.

Example 1: Rescan vSnap server

Assume that you have a vSnap server ({storageId} 2101).

A Python snippet that is similar to the following example can be used to rescan the vSnap server:

```
storage_id = "2101"

_params = {"action": "rescan"}

requests.post('https://' + spp_ipv4 + '/api/storage/' + storage_id,
     headers={...}, params=_params, verify=...)
```

You will receive a response with the HTTP status of 200 (OK).

Example 2: Get information about unregistered disks in a vSnap server

Assume that you rescanned a vSnap server ({storageId} 2101) in the previous example, and this vSnap server has an unused, virtual hard disk drive with the capacity of 200 GB.

A Python snippet that is similar to the following example can be used to get information about

```
storage_id = "2101"

requests.get('https://' + spp_ipv4 + '/api/storage/' + storage_id
     + '/management/disk'
     headers={...}, params=_params, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links":{...},
    "total":1,
    "disks": [
        {
            "links":           {...},
            "resourceType":    "disk",
            "model":           "Virtual disk",
            "name":            "/dev/sdd",
```

```
                "size":             214748364800,
                "type":             "SCSI",
                "uuid":             "1234567890abcdef1234567890abcdef",
                "vendor":           "VMware",
                "usedAs":           "unused",
                "id":               "1234567890abcdef1234567890abcdef",
                "storageId":        "2101",
                "storageServerId": "2101",
                "originalPk":       null,
                "protectionTime":   null,
                "sessionId":        null,
                "name":             "/dev/sdd"
            }
        ]
    }
```


Adding a partner: vSnap server

When you have more than one vSnap server deployed, you can establish a partnership between two vSnap servers that are registered in your IBM Spectrum Protect Plus environment by using the REST API.

Method and URI

You can obtain information about registered vSnap servers through using the `storage` endpoint. You can use this GET method to obtain the storage ID of the two vSnap servers that are to be partnered:

```
GET    https://{hostname|IP}/api/storage
```

To get information about a specific vSnap server registered with IBM Spectrum Protect Plus, use a POST method and a URI:

```
POST    https://{hostname|IP}/api/storage/{storageId}
```

Restriction: With this POST method request, you cannot retrieve information about all vSnap servers at once. Typically, the request takes longer than the GET requests that collect vSnap server information from IBM Spectrum Protect Plus.

Parameters

Parameter 1: `partnerStorageId`

Storage ID of a vSnap server.

- **Value:** *`storage_id`*
- **Type:** Integer. Required. Available when you obtain a list of storage IDs from the IBM Spectrum Protect Plus.

Data

None.

Example: Adding a vSnap server as a partner to another vSnap server

Assume that you have a vSnap server (`{storageId}` 2101).

You will need to create a session with the IBM Spectrum Protect Plus server. Then, get information about all vSnap servers using cached records in IBM Spectrum Protect Plus. Use a GET method. A Python snippet that is similar to the following example can be used to obtain a list of registered vSnap servers:

```
requests.get('https://' + spp_ipv4 + '/api/storage', headers={..}, verify=...)
```

In the response, make note of the storage IDs for the vSnaps. These will be used to create the call to establish the partnership between two servers. Select one of the two vSnap server storage IDs and assign

it to the `storage_id` variable. Use the other storage ID in the `_params` variable. A Python snippet that is similar to the following example can be used to create the partnership between two vSnap using a POST request:

```
storage_id = "2101"

_params = {"partnerStorageId": "2102"}

requests.post('https://' + spp_ipv4 + '/api/storage/' + storage_id + '/management/partner?
action=add',
      headers={...}, params=_params, verify=...)
```

Unregistering a vSnap server

Unregister a vSnap server from IBM Spectrum Protect Plus.

Warning: This operation permanently deletes all backup instances that are stored on the storage resources of the target vSnap server.

Method and URI

To unregister a vSnap server, use a DELETE method with a URI:

```
DELETE  https://{hostname|IPv4}/api/storage/{storageId}
```

Note: To get a {storageId} value based on its name, follow the information in “Getting a {storageId}” on page 242.

Parameters

None.

Data

None.

Example: Unregister a specific vSnap server

Assume that you have a vSnap server, 10.0.1.1 (`{storageId 2001}`), and you want to delete the vSnap server. Run the following command:

```
storage_id = "2101"

requests.delete('https://' + spp_ipv4 + '/api/storage/' + storage_id,
    headers={...}, verify=...)
```

You will receive a response with the HTTP status of 204 (No Content).

Backup Storage: Object Storage

IBM Spectrum Protect Plus supports four types of cloud object storage for backup snapshot data: Amazon Simple Storage Service (S3), IBM Cloud Object Storage, Microsoft Azure Blob Storage, and S3 compatible object storage.

Getting information about object storage accounts and repository servers

You can get a list of object storage servers and repository servers and obtain information about each account and server. Or, you can specify one object storage account or repository server and obtain information about the specified account or server.

Method and URI

To get information about all object storage servers and repository servers, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/cloud
```

To get information about a specific object storage server or repository server, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/cloud/{cloudProviderInfoId}
```

Note: To get a {cloudProviderInfoId} value based on the name of the cloud provider, follow the information in “Getting a {cloudProviderInfoId}” on page 225.

Parameters

None.

Example: Get information about all object storage accounts and repository servers

A Python snippet that is similar to the following example can be used to request object storage and repository server information:

```
requests.get('https://' + spp_ipv4 + '/api/cloud',
    headers={...}, verify=...)
```

The following example response from the Python snippet shows information about object storage and repository servers:

```
{
    "links": {...},
    "total": 6,
    "clouds": [
        {
            "links": {
                "self": {
                    ...
                    "href": "https://10.0.0.100/api/cloud/1",
                    ...
                },
             ...
             },

            "id": "1",
            "name": "S3 Offload Repo",
            "type": "s3",
            "comment": null,
            "accesskey": {
                "href": "https://10.0.0.100/api/identity/key/1003"
            },
```

```
            "properties": {
                "type": "s3",
                "certificate": {
                    "href": "https://10.0.0.101/api/security/certificate/1005"
                },
                "endpoint": "https://demo.s3provider.com.invalid",
                "bucket": "bluemachines",
                "archiveBucket": "bluemachines-archive"
            },
            "provider": "generic",
            "wormProtected": false,
            "defaultRetention": 0,
            "offloadEnabled": true,
            "archiveEnabled": true,
            "ec2ServiceEnabled": null,
            "deepArchiveEnabled": false,
            "rbacPath": "root:0/cloud:0/cloudType:s3/cloud:2",
            "cloudCapacity": null
        },
        {
            "links": {...},
            "id": "2",
            "name": "BlueMachines - IBM COS - Singapore",
            "type": "s3",
            "comment": null,
            "accesskey": {
                "href": "https://10.0.0.100/api/identity/key/1002"
            },
            "properties": {
                "type": "s3",
                "endpoint": "https://ibmcos.bluemachines.com.invalid",
                "bucket": "00000000-0000-0000-0000-000000000000",
                "protectionConfiguration": {
                    "status": "COMPLIANCE",
                    "minimumRetention": 0,
                    "maximumRetention": 2,
                    "defaultRetention": 1
                }
            },
            "provider": "IBM Cloud Object Storage",
            "wormProtected": true,
            "defaultRetention": 1,
            "offloadEnabled": true,
            "archiveEnabled": false,
            "ec2ServiceEnabled": null,
            "deepArchiveEnabled": false,
            "rbacPath": "root:0/cloud:0/cloudType:s3/cloud:1",
            "cloudCapacity": null
        },
        {
            "links": {...},
            "id": "3",
            "name": "BlueMachines - Microsoft Azure - East US",
            "type": "s3",
            "comment": null,
            "accesskey": {
                "href": "https://10.0.0.100/api/identity/key/1003"
            },
            "properties": {
                "type": "s3",
                "endpoint": "https://ibmcos.bluemachines.com.invalid",
                "bucket": "00000000-0000-0000-0000-000000000000",
                "protectionConfiguration": {
                    "status": "COMPLIANCE",
                    "minimumRetention": 0,
                    "maximumRetention": 2,
                    "defaultRetention": 1
                }
            },
            "provider": "Microsoft Azure Blob Storage",
            "wormProtected": true,
            "defaultRetention": 1,
            "offloadEnabled": true,
            "archiveEnabled": false,
            "ec2ServiceEnabled": null,
            "deepArchiveEnabled": false,
            "rbacPath": "root:0/cloud:0/cloudType:s3/cloud:1",
            "cloudCapacity": null
        },
        {
            "links": {...},
            "id": "4",
```

```
            "name": "BlueMachines Amazon S3 US-East1",
            "type": "s3",
            "comment": null,
            "accesskey": {
                "href": "https://10.0.0.100/api/identity/key/1003"
            },
            "properties": {
                "type": "s3",
                "endpoint": "https://ibmcos.bluemachines.com.invalid",
                "bucket": "00000000-0000-0000-0000-000000000000",
                "protectionConfiguration": {
                    "status": "COMPLIANCE",
                    "minimumRetention": 0,
                    "maximumRetention": 2,
                    "defaultRetention": 1
                }
            },
            "provider": "Amazon S3",
            "wormProtected": true,
            "defaultRetention": 1,
            "offloadEnabled": true,
            "archiveEnabled": false,
            "ec2ServiceEnabled": null,
            "deepArchiveEnabled": false,
            "rbacPath": "root:0/cloud:0/cloudType:s3/cloud:1",
            "cloudCapacity": null
        },
        {
            "links": {...},
            "id": "5",
            "name": "Amazon S3-Compatible Storage - Dallas",
            "type": "s3",
            "comment": null,
            "accesskey": {
                "href": "https://10.0.0.100/api/identity/key/1003"
            },
            "properties": {
                "type": "s3",
                "endpoint": "https://ibmcos.bluemachines.com.invalid",
                "bucket": "00000000-0000-0000-0000-000000000000",
                "protectionConfiguration": {
                    "status": "COMPLIANCE",
                    "minimumRetention": 0,
                    "maximumRetention": 2,
                    "defaultRetention": 1
                }
            },
            "provider": "Generic",
            "wormProtected": true,
            "defaultRetention": 1,
            "offloadEnabled": true,
            "archiveEnabled": false,
            "ec2ServiceEnabled": null,
            "deepArchiveEnabled": false,
            "rbacPath": "root:0/cloud:0/cloudType:s3/cloud:1",
            "cloudCapacity": null
        },
        {
            "links": {...},
            "id": "6",
            "name": "BlueMachines - IBM Spectrum Protect - Dallas",
            "type": "s3",
            "comment": null,
            "accesskey": {
                "href": "https://10.0.0.100/api/identity/key/1015"
            },
            "properties": {
                "type": "s3",
                "certificate": {
                    "href": "https://10.0.0.100/api/security/certificate/1004"
                },
                "hostname": "DUMMY-SP01",
                "port": 9000.0,
                "bucket": "spp-spp10-1000000000000"
            },
            "provider": "sp",
            "wormProtected": false,
            "defaultRetention": 0,
            "offloadEnabled": true,
            "archiveEnabled": true,
            "ec2ServiceEnabled": null,
            "deepArchiveEnabled": false,
```

```
            "rbacPath": "root:0/cloud:0/cloudType:s3/cloud:1",
            "cloudCapacity": null
        }
    ]
}
```

Getting bucket information

You can get information about buckets that are associated with an object storage account. Bucket information is required to register an object storage account with IBM Spectrum Protect Plus.

Method and URI

To get bucket information, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/api/cloud
```

Parameters

Parameter 1: `action`

Request information about object storage buckets that are associated with cloud storage accounts.

- **Value:** `getBuckets`
- **Type:** System string. Required. Available in the web user interface.

Data

Data 1: `provider`

The type of cloud storage provider. Use one of the following values:

Provider	Value
Amazon S3	aws
IBM Cloud Object Storage	cos
Microsoft Azure Blob Storage	azure
S3 compatible object storage	generic

- **Type:** System string. Required. Available in the web user interface.

Data 2: `accesskey`

An access key that was registered with IBM Spectrum Protect Plus. For details, follow the instructions in “Keys and Certificates” on page 140. To get the bucket information about a new object storage account, enter an empty JSON object (`{}`).

- **Example value:**

```
{
    "href": "https://10.0.0.100/api/identity/key/1003"
}
```

- **Type:** JSON object.

Data 2.1: `accesskey` > `href`

The URL of the access key. Typically, it has the following syntax:

```
https://{hostname|IPv4}/api/identity/key/{identityKeyId}
```

- **Type:** String.

Note: To get an {identityKeyId} value based on the name of the identity key, follow the information in "Getting an {identityKeyId}" on page 233.

Data 3: `properties`

The properties of the object storage account.

- **Type:** Array. Required. Available in the web user interface.

Data 3.1: `properties` > `region`

The region of the Amazon S3 storage account. Use one of the following values:

Amazon S3 region	Value
US East (Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Asia Pacific (Tokyo)	ap-northeast-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Osaka-Local)	ap-northeast-3
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Canada (Central)	ca-central-1
China (Beijing)	cn-north-1
China (Ningxia)	cn-northwest-1
EU (Frankfurt)	eu-central-1
EU (Ireland)	eu-west-1
EU (London)	eu-west-2
EU (Paris)	eu-west-3
EU (Stockholm)	eu-north-1
South America (São Paulo)	sa-east-1

- **Type:** System string.

Data 3.2: `properties` > `endpoint`

The endpoint URL of the account managed by IBM Cloud Object Storage, S3 compatible object storage, or Microsoft Azure Blob Storage. For Microsoft Azure Blob Storage accounts, use one of the following endpoint URLs:

Microsoft Azure endpoint	Value
Microsoft Azure Global	https://core.windows.net:443
Microsoft Azure Germany	https://core.cloudapi.de:443
Microsoft Azure China 21 Vianet	https://core.chinacloudapi.cn:443

Microsoft Azure endpoint	Value
Microsoft Azure USA Government	https://core.usgovcloudapi.net:443

- **Type:** String.

Data 3.3: `properties` > `accessKey`

The access key of the account managed by Amazon S3, IBM Cloud Object Storage, or S3 compatible object storage, or the storage account name of the Microsoft Azure Blob Storage account.

- **Example value:** `QAH6fG85SkPmyEfqn4yy`
- **Type:** String

Data 3.4: `properties` > `secretKey`

The secret access key for an Amazon S3 or S3 compatible object storage account; the secret key for an IBM Cloud Object Storage account, or the storage account shared key, account key, or access key for a Microsoft Azure Blob Storage account.

Example 1: Get bucket information for an Amazon S3 account

Assume that you want to get information about buckets that are associated with the following Amazon S3 account:

- **Region:** US West (N. California)
- **Access key:** `Q3Z3MCDF7WJVG7KREFWFG`
- **Secret access key:** `rNzSHk2gBfjf9cECGp+cjrhEdhsZmxwHnRckk8BP`

A Python snippet that is similar to the following example can be used to get information about the buckets that are associated with this Amazon S3 account:

```
_params = {"action": "getBuckets"}

_data = f'''{{
    "provider":    "aws",
    "properties":  {{
        "region":    "us-west-1",
        "accessKey": "Q3Z3MCDF7WJVG7KREFWFG",
        "secretKey": "rNzSHk2gBfjf9cECGp+cjrhEdhsZmxwHnRckk8BP"
    }}
}}'''

requests.post('https://' + spp_ipv4 + '/api/cloud',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

Tip: Every bucket object has the name and ID that are the same string.

```
{
    "total": 3,
    "buckets": [
        {
            "links": {},
            "id": "bluemachines-sales",
            "name": "bluemachines-sales",
            "protectionConfiguration": null,
            "haslifecycleRules": false,
            "hasBucketArchiveLifecycleRules": false,
            "resourceType": "bucket"
        },
        {...}, {...}
    ]
}
```

Example 2: Get bucket information by using an existing access key

As mentioned, you can also reuse an existing access key instead of entering the access key and the secret key of the object storage account. Assume that you want to get information about buckets that are associated with the following Amazon S3 account and the access key that was registered with IBM Spectrum Protect Plus:

- **Region:** US West (N. California)
- **Access key:** BlueMachines Amazon S3 California Key (`{identityKeyId}` 1003)

A Python snippet that is similar to the following example can be used to get information about the buckets that are associated with this Amazon S3 account:

```
_params = {"action": "getBuckets"}

_data = f'''{{
    "provider":    "aws",
    "accesskey":   {{
        "href": "https://10.0.0.100/api/identity/key/1003"
    }}
    "properties":  {{
        "region":    "us-west-1",
    }}
}}'''

requests.post('https://' + spp_ipv4 + '/api/cloud',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response with the HTTP status of 200 (OK). The response body is similar to the JSON object, as shown in “Example 1: Get bucket information for an Amazon S3 account” on page 111.

```
{
    "total": 3,
    "buckets": [
        {
            "links": {},
            "id": "bluemachines-sales",
            "name": "bluemachines-sales",
            "protectionConfiguration": null,
            "haslifecycleRules": false,
            "hasBucketArchiveLifecycleRules": false,
            "resourceType": "bucket"
        },
        {...}, {...}
    ]
}
```

Example 3: Get bucket information for an IBM Cloud Object Storage account

Assume that you want to get information about buckets that are associated with the following IBM Cloud Object Storage account:

- **Endpoint:** https://ibmcos.bluemachines.com.invalid
- **Access key:** `QAH6fG85SkPmyEfqn4yy`
- **Secret access key:** `ddUTAun9DBHuehkC2TLwkfH7WyEzQ8PaCxJ4xNYn`

A Python snippet that is similar to the following example can be used to get information about the buckets that are associated with this IBM Cloud Object Storage account:

```
_params = {"action": "getBuckets"}

_data = f'''{{
    "provider":    "cos",
    "properties":  {{
        "endpoint":  "https://ibmcos.bluemachines.com.invalid",
        "accessKey": "QAH6fG85SkPmyEfqn4yy",
        "secretKey": "ddUTAun9DBHuehkC2TLwkfH7WyEzQ8PaCxJ4xNYn"
    }}
```

```
}}'''

requests.post('https://' + spp_ipv4 + '/api/cloud',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response with the HTTP status of 200 (OK). The response body is similar to the JSON object, as shown in “Example 1: Get bucket information for an Amazon S3 account” on page 111.

Example 4: Get bucket information for a Microsoft Azure Blob Storage account

Assume that you want to get information about buckets that are associated with the following Microsoft Azure Blob Storage account:

- **Endpoint:** Microsoft Azure Global
- **Storage Account Name:** bluemachines-azure
- **Storage Account Shared Key:** VUAZ87JC6MKT7D5FXW+YZ24DDXXHWP3P9HZT +2J5V88LN8KWAZH4NW3R4PE495FHWKVZPTPSGR4QEUJ5TWH4DL==

A Python snippet that is similar to the following example can be used to get information about the buckets that are associated with this Microsoft Azure Blob Storage account:

```
_params = {"action": "getBuckets"}

_data = f'''{{
    "provider":    "azure",
    "properties":  {{
        "endpoint":  "https://core.windows.net:443",
        "accessKey": "azure-bluemachines",
        "secretKey": "VUAZ87JC6MKT7D5FXW+YZ24DDXXHWP3P9HZT+2J5V88LN8KWAZH4NW3R4P
   E495FHWKVZPTPSGR4QEUJ5TWH4DL=="
    }}
}}'''

requests.post('https://' + spp_ipv4 + '/api/cloud',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response with the HTTP status of 200 (OK). The response body is similar to the JSON object, as shown in “Example 1: Get bucket information for an Amazon S3 account” on page 111.

Example 5: Get bucket information about an S3 compatible object storage account

Assume that you want to get information about buckets that are associated with the following S3 compatible object storage account:

- **Endpoint:** https://s3.volvox.com.invalid
- **Access key:** PJNG6J5FHENHWZ66K77X
- **Secret access key:** CFBF8QcKtMSMYTE++mX3pUWSwmchNBgg5afbnw5T

A Python snippet that is similar to the following example can be used to get information about the buckets that are associated with this the S3 compatible object storage account:

```
_params = {"action": "getBuckets"}

_data = f'''{{
    "provider":    "generic",
    "properties":  {{
        "endpoint":  "https://s3.volvox.com.invalid",
        "accessKey": "PJNG6J5FHENHWZ66K77X",
        "secretKey": "CFBF8QcKtMSMYTE++mX3pUWSwmchNBgg5afbnw5T"
    }}
}}'''
```

```
requests.post('https://' + spp_ipv4 + '/api/cloud',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response with the HTTP status of 200 (OK). The response body is similar to the JSON object, as shown in “Example 1: Get bucket information for an Amazon S3 account” on page 111.

Adding an object storage server or a repository server

You can add an object storage server (Amazon S3, Azure Blog Object Storage, IBM Cloud Object Storage, or S3 compatible storage) or a repository server (IBM Spectrum Protect).

Method and URI

To add an object storage server, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/ngp/cloud
```

Parameters

None.

Data

Data 1: `type`

- **Value:** `s3`
- **Type:** System string. Required.

Data 2: `provider`

The type of cloud storage provider. For available key-value pairs, follow the instructions in “Getting bucket information” on page 109.

Provider	Value
Amazon S3	aws
IBM Cloud Object Storage	cos
Microsoft Azure Blob Storage	azure
S3 compatible object storage	generic
IBM Spectrum Protect	sp

- **Example value:** `cos`
- **Type:** System string. Required. Available in the web user interface.

Data 3: `accesskey`

The URL of the access key. The URL has the following format:

```
https://{hostname|IPv4}/api/identity/key/{identityKeyId}
```

Tip: To get an {identityKeyId} value based on the name of the identity key, follow the information in “Getting an {identityKeyId}” on page 233.

- **Example value:** https://10.0.0.100/api/identity/key/1010
- **Type:** String. Required. Available in the web user interface.

Data 4: `properties`

- **Example value:**

```
{
    "type":          "s3",
    "certificate":   "10.0.0.100/api/security/certificate/1287",
    "endpoint":      "https://ibmcos.bluemachines.com.invalid",
    "bucket":        "bluemachines-sales",
    "archiveBucket": "bluemachines-archive"
}
```

- **Type:** JSON object. Required. Available in the web user interface.

Data 4.1: `properties` > `type`

- **Value:** `s3`
- **Type:** System string. Required.

Data 4.2: `properties` > `certificate`

The URL of the certificate for IBM Cloud Object Storage and S3 compatible object storage: `{certificateHref}`.

Tip: To get a {certificateId} value based on the certificate name, follow the information in "Getting a {certificateId}" on page 239.

- **Type:** String. Available in the web user interface.

Data 4.3: `properties` > `region`

The region for Amazon S3. For available key-value pairs, follow the instructions in "Getting bucket information" on page 109.

- **Example value:** `us-east-1`
- **Type:** String. Available in the web user interface.

Data 4.4: `properties` > `enableDeepArchive`

Enable or disable the Amazon S3 Glacier Deep Archive storage class for Amazon S3.

- **Example value:** false
- **Type:** Boolean. Available in the web user interface.

Data 4.5: `properties` > `endpoint`

The URL of the endpoint of the object storage server.

- **Type:** String. Required. Available in the web user interface.

Data 4.6: `properties` > `bucket`

The name of the standard object storage bucket.

- **Type:** String. Required. Available in the web user interface.

Tip: To get the bucket name, follow the information in "Getting bucket information" on page 109.

Data 4.7: `properties` > `archiveBucket`

The name of the archive storage object bucket.

- **Type:** String. Available in the web user interface.

Data 4.8: `properties` > `hostname`

The hostname of IBM Spectrum Protect.

- **Example value:** sp-dallas1
- **Type:** String. Required. Available in the web user interface.

Data 4.9: `properties` > `port`

The port number to connect to IBM Spectrum Protect.

- **Example value:** 9000
- **Type:** Integer. Required. Available in the web user interface.

Data 5: `name`

The name of the object storage server or the repository server.

- **Example value:** BlueMachines - IBM COS - Singapore
- **Type:** String. Required. Available in the web user interface.

Example 1: Add an IBM Cloud Object Storage server

Assume that you want to add the following IBM Cloud Object Storage server:

- **Provider:** IBM Cloud Object Storage
- **Name:** BlueMachines - IBM COS - Dallas
- **Access key:** BlueMachines IBM COS Dallas Key (`{identityKeyId}` 1010)
- **Certificate (optional):** BlueMachines - Cert IBM Cloud Object Storage (`{certificateId}` 1287)
- **Standard object storage bucket:** bluemachines-sales
- **Archive object storage bucket:** bluemachines-archive

A Python snippet that is similar to the following example can be used to send a request to IBM Spectrum Protect Plus to add the object storage server:

```
_data = f'''{{
    "type":       "s3",
    "provider":   "cos",
    "accesskey":  "https://10.0.0.100/api/identity/key/1010",
    "properties": {{
        "type":          "s3",
        "certificate":   "10.0.0.100/api/security/certificate/1287",
        "endpoint":      "https://ibmcos.bluemachines.com.invalid",
        "bucket":        "bluemachines-sales",
        "archiveBucket": "bluemachines-archive"
    }},
"name":           "BlueMachines - IBM COS - Dallas"
}}'''

requests.post('https://' + spp_ipv4 + '/ngp/cloud',
    headers={...}, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 201 (Created) and verify that the object storage server was added. The response body is similar to the JSON object for the object storage server, as shown in "Getting information about object storage accounts and repository servers" on page 106.

Example 2: Add IBM Spectrum Protect

Assume that you want to add the following IBM Spectrum Protect system:

- **Name:** BlueMachines Spectrum Protect - Dallas 1
- **Hostname:** sp-dallas1
- **Port:** 9000
- **Access key:** BlueMachines SP Dallas1 Key (`{identityKeyHref}` https://10.0.0.100/api/identity/key/1050)
- **Certificate (optional):** BlueMachines - Cert IBM Spectrum Protect (`{certificateHref}` https://10.0.0.100/api/security/certificate/1287)

A Python snippet that is similar to the following example can be used to send a request to IBM Spectrum Protect Plus to add the IBM Spectrum Protect server:

```
access_key_href = "https://10.0.0.100/api/identity/key/1050"
certificate_key_href = "https://10.0.0.100/api/security/certificate/1287"

_data = f'''{{
    "type":       "s3",
    "provider":   "sp",
    "accesskey":  {access_key_href},
    "properties": {{
        "type":        "s3",
        "certificate": {certificate_key_href}
        "hostname": sp-dallas1
        "port":        9000
    }},
    "name":       "BlueMachines - IBM COS - Dallas"
}}'''

requests.post('https://' + spp_ipv4 + '/ngp/cloud',
    headers={...}, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 201 (Created) and the IBM Spectrum Protect was added. The response body is similar to the JSON object for the IBM Spectrum Protect, as shown in “Getting information about object storage accounts and repository servers” on page 106.

Unregistering an object storage server or a repository server

You can unregister an object storage server or a repository server from IBM Spectrum Protect Plus.

Method and URI

To unregister an object storage server or a repository server, use a DELETE method and a URI:

```
DELETE  https://{hostname|IPv4}/api/api/cloud/{cloudProviderInfoId}
```

Tip: To get an {cloudProviderInfoId} value based on its name, follow the information in “Getting a {cloudProviderInfoId}” on page 225.

Parameters

None.

Data

None.

Example: Unregister an Amazon S3 object storage server

Assume that you want to unregister an Amazon S3 object storage server, BlueMachines Amazon S3 US-East1 (`{identityKeyId}` 4). Run the following command:

```
cloud_provider_info_id = "4"

requests.delete('https://' + spp_ipv4 + '/api/cloud/' + cloud_provider_info_id,
    headers={...}, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 204 (No Content) and that you no longer see the cloud object storage server. You can unregister other types of object storage servers (Amazon S3, IBM Cloud Object Storage, and S3 compatible object storage) and repository servers (IBM Spectrum Protect) in a similar way.

Backup Storage: Repository Server

IBM Spectrum Protect Plus supports one type of repository servers for backup storage: IBM Spectrum Protect.

Getting repository server information

You can get a list of repository servers registered in IBM Spectrum Protect Plus and each of them.

Method and URI

To get information about all repository servers, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/cloud
```

To get information about a specific repository server, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/cloud/{cloudProviderInfoId}
```

Tip: To get an {cloudProviderInfoId} value based on its name, follow the information in “Getting a {cloudProviderInfoId}” on page 225.

Parameters

None.

Data

None.

Example 1: Get information about all repository servers

The following Python snippet requests a list of all repository servers and their information:

```
requests.get('https://' + spp_ipv4 + '/api/cloud',
    headers={...}, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "total": 2,
    "clouds": [
        {
            "links": {...},
            "id": "1",
            "name": "BlueMachines - Spectrum Protect - Singapore 1",
            "type": "s3",
            "comment": null,
            "accesskey": {
                "href": "https://10.0.0.100/api/identity/key/1001"
            },
            "properties": {
                "type": "s3",
                "certificate": {
                    "href": "https://10.0.0.100/api/security/certificate/1001"
                },
                "hostname": "SP-SINGAPORE1",
                "port": 9000,
                "bucket": "spp-singapore1-1575895669224"
            },
            "provider": "sp",
            "wormProtected": false,
            "defaultRetention": 0,
```

```
                "offloadEnabled": true,
                "archiveEnabled": true,
                "ec2ServiceEnabled": null,
                "deepArchiveEnabled": false,
                "rbacPath": "root:0/cloud:0/cloudType:s3/cloud:1",
                "cloudCapacity": null,
                "name": "BlueMachines - Spectrum Protect - Singapore 1",
                "id": "1",
                "type": "s3"
            },
            {
                ...
                "id": "2,
                "name": "BlueMachines - Spectrum Protect - Dallas 1",
                ...
            }
        ]
    }
```

Example 2: Get information about a specific repository server

Assume that you want to get information about the specific repository server. You get the cloudProviderInfoId of it: 2. A Python snippet that is similar to the following example can be used to request the information about this repository server:

```
cloud_provider_info_id = "2"

requests.get('https://' + spp_ipv4 + '/api/cloud'
    + cloud_provider_info_id,
    headers={...}, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    ...
    "id": "2,
    "name": "BlueMachines - Spectrum Protect - Dallas 1",
    ...
}
```

Adding a repository server

To register a repository server, follow the instructions in “Adding an object storage server or a repository server” on page 114.

Unregistering a repository server

You can delete a repository server you have registered. *It must be unused by IBM Spectrum Protect Plus.*

Method and URI

To delete a repository server with `{cloudProviderInfoId}`, the request's method and URI are similar to the following example:

```
DELETE  https://{hostname|IPv4}/api/cloud/{cloudProviderInfoId}
```

Tip: To get an {cloudProviderInfoId} value based on its name, follow the information in “Getting a {cloudProviderInfoId}” on page 225.

Parameters

None.

Data

None.

Example

Assume that you have a repository server BlueMachines - Spectrum Protect - Dallas 1 and you want to unregister it. You get the `{cloudProviderInfoId}` of it: 2. Run the command below.

```
cloud_provider_info_id = "2"

requests.delete('https://' + spp_ipv4 + '/api/cloud/'
    + cloud_provider_info_id,
    headers={...}, params="", data="", verify=...)
```

Ensure you get a response with the HTTP status of 204 (No Content) and that you no longer see the repository server BlueMachines - Spectrum Protect - Dallas 1.

VADP proxy

IBM Spectrum Protect Plus supports VMware vStorage API for Data Protection (VADP) to load-balance data traffic when it is operating on a VMware environment.

VADP proxies can be installed on vSnap virtual appliances or you can also install and use a VADP proxy server on a supported Linux virtual machine. You must have at least one VADP proxy in the site if it is a VMware environment.

Getting VADP proxy information

You can get information about VADP proxies that are registered with IBM Spectrum Protect Plus.

Method and URI

To get information about all VADP proxy servers, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/vadp
```

Parameters

None.

Data

None.

Example: Get VADP proxy information

Assume that you registered a VADP proxy with IBM Spectrum Protect Plus.

A Python snippet that is similar to the following example can be used to get information about the VADP proxies:

```
requests.get('https://' + spp_ipv4 + '/api/vadp',
    headers={...}, verify=...)
```

The following response body displays information about VADP proxies. The HTTP status of this response is 200 ("OK").

```
{
    "links": {...},
    "vadps": [
        {
            "links": {...},
            "id": 2103,
            "siteId": 3101,
            "siteName": "Dallas",
            "displayName": "centos-vadp-dallas1",
            "hostName": "10.0.2.1",
            "ipAddr": "10.0.2.1",
            "securePort": 8098,
            "rbacPath": "root:0/proxy:0/proxy.type:VMDKBACKUP.VADPPROXY/proxy.ty
   pe.VMDKBACKUP.VADPPROXY:2,103",
            "corecount": 0,
            "availablememory": 0,
            "version": "10.1.8.800",
            "state": "ENABLED",
            "stateDisplayName": "Enabled",
            "isLocal": false,
            "demo": false,
            "user": {
                "href": "https://10.0.0.100/api/identity/user/2154"
            }
        },
        {
            "links": {...},
            "id": 2000,
            "siteId": 1000,
            "siteName": "Dallas",
            "displayName": "localhost",
            "hostName": "localhost",
            "ipAddr": "127.0.0.1",
            "port": 8080,
            "rbacPath": "root:0/proxy:0/proxy.type:VMDKBACKUP.VADPPROXY/proxy.ty
   pe.VMDKBACKUP.VADPPROXY:2,000",
            "corecount": 0,
            "availablememory": 0,
            "version": "10.1.8.800",
            "state": "SUSPENDED",
            "stateDisplayName": "Suspended",
            "isLocal": true,
            "demo": false,
            "user": {
                "href": "https://10.0.0.100/api/identity/user/-1"
            }
        },
        {...}, {...}
    ],
    "enabled": [
        {
            "links": {...},
            "id": 2000,
            "siteId": 1000,
            "siteName": "Dallas",
            "displayName": "localhost",
            ...
        },
        {
            "links": {...},
            "id": 2103,
            ...,
        },
        {...}
    ],
    "suspended": [
        {
            "links": {...},
            "id": 2105,
            "siteId": 1001,
            "siteName": "Singapore",
            "displayName": "vsnap-singapore1",
            ...,
            "version": "10.1.5.1008",
            "state": "SUSPENDED",
            "stateDisplayName": "Suspended",
```

```
            ...
        }
    ],
    "available": []
}
```

Registering a VADP proxy

You can register a VADP proxy with the IBM Spectrum Protect Plus system environment.

Method and URI

To register a VADP proxy with the IBM Spectrum Protect Plus system environment, use a POST method and a URI:

```
POST    https://{hostname|IPv4}/ngp/vadp
```

Parameters

None.

Data

Data 1: `pushinstall`

- **Example value:**

Install the VADP proxy application on the Linux system.

```
{
    "hostAddress": "10.0.2.1"
}
```

- **Type:** JSON object. Required. Available in the web user interface.

Data 1.1: `pushinstall` > `hostAddress`

The IPv4 address or the hostname of the Linux system.

Data 2: `identityId`

The `{identityUserId}` of the identity to log in to the Linux system.

- **Example value:** 2110

  Tip: To get a {siteId} value based on the name of the site, follow the information in “Getting a {siteId}” on page 240.

- **Type:** String. Required. Available in the web user interface.

Data 3: `registration`

- **Example value:**

```
{
    "siteId": "3102"
}
```

- **Type:** JSON object. Required. Available in the web user interface.

Data 3.1: `registration` > `siteId`

The `{siteId}` of the site to associate with the VADP proxy.

Tip: To get a {siteId} value based on the name of the site, follow the information in “Getting a {siteId}” on page 240.

Type: String. Required. Available in the web user interface.

Example: Installing a VADP proxy

Assume that you want to install a VADP proxy on the following Linux system and add it to IBM Spectrum Protect Plus:

- **Name:** centos-vadp-dallas1
- **IPv4 address:** 10.0.2.1
- **Identity:** sarah-vadp1_10.0.2.1 (`{identityUserId}` 2110)
- **Site:** Dallas (`{siteId}` 3101)

A Python snippet that is similar to the following example can be used to install the VADP proxy on the Linux system and register the proxy with IBM Spectrum Protect Plus:

```
_data = f'''{{
     "pushinstall":  {
         "hostAddress": "10.0.2.1"
     },
     "identityId":   "2110",
     "registration": {
         "siteId":   "3101"
     }

}}'''

requests.post('https://' + spp_ipv4 + '/ngp/vadp',
    headers={...}, data=_data, verify=...)
```

The request prompts a response with the HTTP status of 201 (Created). Review the response to ensure that the new VADP proxy, centos-vadp-dallas1, was added. The response body is similar to the JSON object for centos-vadp-dallas1, as shown in "Getting VADP proxy information" on page 120.

Additional actions for a VADP proxy

You can take additional actions with a VADP proxy: suspending the VADP proxy temporarily, resuming the suspended VADP proxy, and uninstalling the VADP proxy application from the VADP proxy server. If you uninstall the VADP proxy will be unregistered from IBM Spectrum Protect Plus.

Method and URI

To take additional actions such as suspending, resuming, or uninstalling the VADP proxy, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/api/vadp/{vadpId}
```

Tip: To get a {vadpId} value based on the IPv4 address of the VADP proxy, for more information see "Getting a {vadpId}" on page 243.

Parameters

Parameter 1: `action`

Specify the type of action.

- **Value:** Use one of the following values:

Action	Value
Suspend	suspend
Resume	resume

Action	Value
Uninstall	uninstall

Note: To unregister a VADP proxy, follow the information in “Unregistering a VADP proxy” on page 124.

- **Type:** System string. Required. Available in the web user interface.

Data

None.

Example: Suspend a VADP proxy

Assume that you have a VADP proxy, centos-vadp-dallas1 ({vadp_id} 2102), and you want to suspend the VADP proxy.

A Python snippet that is similar to the following example can be used to suspend the VADP proxy from IBM Spectrum Protect Plus:

```
_params = {"action": "suspend"}

requests.post('https://' + spp_ipv4 + '/api/vadp/' + vadp_id,
    headers={...}, params=_params, verify=...)
```

Ensure that you get a response with the HTTP status of 200 (OK) and the VADP proxy is suspended. You can take other actions (resuming and uninstalling) in a similar way.

Unregistering a VADP proxy

You can unregister a VADP proxy from the IBM Spectrum Protect Plus.

Method and URI

To unregister a VADP proxy with {vadpId}, use the DELETE method with a URI:

```
DELETE  https://{hostname|IPv4}/api/vadp/{vadpId}
```

Tip: To get a {vadpId} value based on the IPv4 address of the VADP proxy, follow the information in “Getting a {vadpId}” on page 243 .

Parameters

None.

Data

None.

Example: Unregister a VADP proxy

Assume that you have a VADP proxy, centos-vadp-singapore1 ({vadp_id} 2103), and you want to unregister the VADP proxy.

A Python snippet that is similar to the following example can be used to unregister the VADP proxy from IBM Spectrum Protect Plus:

```
vadp_id = "2103"

requests.delete('https://' + spp_ipv4 + '/api/vadp/' + vadp_id,
    headers={...}, data=_data, verify=...)
```

Ensure that you get a response with the HTTP status of 204 (No Content) and that you no longer see the VADP proxy centos-vadp-singapore1. The response body is empty.

Site

A *site* is used to group vSnap servers based on a physical or logical location to manage data placement in the environment. A site is assigned to a vSnap server when the server is added to IBM Spectrum Protect Plus.

Sites are not applicable to object storage (for example, IBM Cloud Object Storage or Amazon S3) or to repository servers (IBM Spectrum Protect).

Getting site information

You can get a list of sites and information for each, or get information about a specific site.

Method and URI

To get information about all sites, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/site
```

To get information about a specific site, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/site/{siteId}
```

Tip: To get a {siteId} value based on its name, following the information in “Getting a {siteId}” on page 240.

Parameters

None.

Data

None.

Example 1: Get information about all sites

Assume that you have added a site, "Dallas". A Python snippet that is similar to the following example can be used to request a list of all sites and their information:

```
requests.get('https://' + spp_ipv4 + '/api/site',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "total": 4,
    "sites": [
        {
            "links": {...},
            "id": "2000",
            "name": "Secondary",
            "description": "secondary site",
            "defaultSite": false,
            "throttles": null,
            "demo": false,
            "rbacPath": "root:0/site:0/site:2000"
        },
        {
```

```
            "links": {...},
            "id": "1000",
            "name": "Primary",
            "description": "Primary Site",
            "defaultSite": true,
            "throttles": null,
            "demo": false,
            "rbacPath": "root:0/site:0/site:1000"
        },
        {
            "links": {...},
            "id": "3110",
            "name": "Dallas",
            "description": null,
            "defaultSite": false,
            "throttles": [
                {
                    "rate": "786432000",
                    "schedules": [
                        {
                            "start": 133200,
                            "end": 165599
                        },
                        {
                            "start": 219600,
                            "end": 251999
                        },
                        ...,
                        {
                            "start": 565200,
                            "end": 590399
                        }
                    ]
                }
            ],
            "demo": false,
            "rbacPath": "root:0/site:0/site:3110"
        }
    ]
}
```

For information about enabling throttling for sites to limit the bandwidth used by IBM Spectrum Protect Plus, follow the instructions in “Adding a site” on page 127.

Example 2: Get information about a specific site

Assume that you want to get information about the site with the following `{siteId}`: 2000. A Python snippet that is similar to the following example can be used to request the information about this site:

```
site_id = "2000"

requests.get('https://' + spp_ipv4 + '/api/site/' + site_id,
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "id": "2000",
    "name": "Secondary",
    "description": "secondary site",
    "defaultSite": false,
    "throttles": null,
    "demo": false,
    "rbacPath": "root:0/site:0/site:2000"
}
```

Adding a site

You can add a new site.

Method and URI

To add a site, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/api/site
```

Parameters

None.

Data

Data 1: `name`

The name of the site.

- **Example value:** Dallas
- **Type:** String. Required. Available in the web user interface.

Data 2: `description`

The description of the site.

- **Example value:** BlueMachines Dallas data center
- **Type:** String.

Data 3: `defaultSite`

Indication of whether the site is set as the default.

Requirement: You must have at least one default site, which is usually Primary. If you set another site as a default site, the current one is unset.

- **Example value:** false
- **Type:** Boolean.

Data 4: `throttles`

The throttle settings.

Restriction: Throttle settings apply to outgoing traffic from vSnap servers in replication, archive, and copy operations, but not to restore operations.

Tip: If you associated a vSnap server with a site, the vSnap server uses the throttle settings of the site with the time zone setting of the vSnap server.

- **Example value:**

```
[
    {
        "rate":      "786432000",
        "schedules": [
            {
                "start": 115200,
                "end":   147599
            },
            {
                "start": 201600,
                "end":   233999
            }
        ]
    }
]
```

- **Type:** List. Available in the web user interface.

Data 4.1: `throttles` > JSON object > `rate`

The throughput in bytes per second.

- **Type:** Integer. Available in the web user interface.

Data 4.2 `throttles` > JSON object > `schedules`

Schedule settings. The sample values, which start at 0 and end at 3600, specify a range beween midnight and 1 AM on Sunday. By default, both values are set to 0. As a result, throttling is disabled.

- **Type:** Array. Available in the web user interface.

Data 4.2.1 `throttles` > JSON object > `schedules` > JSON object > `start`

The start time defined by the number of seconds after Sunday 00:00:00.

Type: Integer. Available in the web user interface.

Data 4.2.2 `throttles` > JSON object > `schedules` > JSON object > `start`

The end time defined by the number of seconds after Sunday 00:00:00.

Type: Integer. Available in the web user interface.

Data 5: `loadBalanceRatio`

VMware VM allocation.

Restriction: Do not enter a value other than 100 unless you are instructed to do so by IBM Software Support.

- **Value:** 100
- **Type:** Integer. Available in the web user interface.

Example: Add a new site with throttle settings

As you can see in the IBM Spectrum Protect Plus interface in your web browser, every site allows you to specify throttles to restrict the bandwidth that IBM Spectrum Protect Plus uses.

For example, assume that you have an SLA policy that takes a backup snapshot every hour. You do not want to consume too much bandwidth with this job during business hours, which are from 8 AM through 5 PM Monday through Friday and 8 AM through 3 PM on Saturdays. Your office is located in the US Central Daylight Time (CDT) zone.

Log in to one of the vSnap servers that are associated with the site, and see the time zone setting of the vSnap server:

```
   [sarah@workstation ~ ] ssh sarah-vsnap1@10.0.1.1    # vsnap-dallas1
   sarah-vsnap1@10.0.1.1's password: 6bxPYs6eQNa!V85R
   Last login: Mon Jun 01 00:00:00 2020 from 10.0.0.1
   ------------------------------------------------------------------
   Be sure to adhere to vSnap hardware and memory requirements
    as described in IBM Spectrum Protect
Plus Blueprints
    accessible from the IBM Spectrum Protect
Plus Knowledge Center.
   ------------------------------------------------------------------
```

```
   [sarah-vsnap1@vsnap-dallas1 ~]$ timedatectl
         Local time: Mon 2020-06-01 01:00:00 UTC
     Universal time: Mon 2020-06-01 01:00:00 UTC
           RTC time: Mon 2020-06-01 01:00:00
          Time zone: UTC (UTC, +0000)
        NTP enabled: yes
   NTP synchronized: yes
    RTC in local TZ: yes
```

```
        DST active: n/a
...
```

This output indicates that the time zone of this vSnap server is set to UTC, which is 5 hours ahead of CDT.

You want to limit the bandwidth to a maximum of 750 MB per second during these business hours and allow unlimited bandwidth after hours.

The rate is 750 MB per second, which is 786,432,000 bytes per second. Monday at 8 AM CDT or 1 PM UTC is 133,200 seconds after 0:00 AM on Sunday. This value can be calculated by using the following formula: 60 * 60 * 24 + 60 * 60 * 13. The other time ranges can be calculated in a similar way.

```
_data = f'''{{
    "name":         "Dallas",
    "description": "BlueMachines Dallas Data Center",
    "defaultSite": false,
    "throttles":   [
        {{
            "rate": "786432000",
            "schedules": [
                {{"start": 133200, "end":165599}},
                {{"start": 219600, "end":251999}},
                {{"start": 306000, "end":338399}},
                {{"start": 392400, "end":424799}},
                {{"start": 478800, "end":511199}},
                {{"start": 565200, "end":590399}}
            ]
        }}
    ],
    "loadBalanceRatio": 100
}}'''

requests.post('https://' + spp_ipv4 + '/api/site',
    headers={...}, data=_data, verify=...)
```

The request prompts a response with the HTTP status of 201 (Created). The response body is similar to the JSON object for the Dallas site, as shown in “Getting site information” on page 125. Review the response to ensure that the new site, Dallas, was created.

Editing the configuration of a site

You can edit settings for a site. You can change the site name and other options for the default Primary and Secondary sites.

Method and URI

To update the site of `{siteId}`, use a PUT method and a URI:

```
PUT     https://{hostname|IPv4}/api/site/{siteId}
```

Tip: To get a {siteId} value based on the name of the site, follow the information in “Getting a {siteId}” on page 240.

Parameters

None.

Data

The data key-value pairs are the same as for a site. For more information, follow the instructions in “Adding a site” on page 127.

Example: Edit the configuration of a site

As you can see in the IBM Spectrum Protect Plus interface in your web browser, every site allows you to specify throttles to restrict the bandwidth that IBM Spectrum Protect Plus uses.

For example, assume that you added a site that is named Dallas (`{siteId}` 3100). You want to assign the following values to this site:

- Rename it Singapore
- Remove all throttles
- Add the description BlueMachines Singapore Data Center.

A Python snippet that is similar to the following example can be used to update the configuration of a site:

```
site_id = "3100"    # Dallas

_data = f'''{{
    "name":         "Singapore",
    "description": "BlueMachines Singapore Data Center",
    "defaultSite": false,
    "throttles":   [],
    "loadBalanceRatio": 100
    }}'''

requests.put('https://10.0.0.100/api/site/' + site_id,
    headers={...}, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 200 (OK) and that the site is updated. This request is similar to the one that you used to create the site; compare the HTTP request method, the URI, the `data` values, and the HTTP status code in the response.

Deleting a site

You can delete a site.

Method and URI

To delete a site with `{siteId}`, use the DELETE method with a URI:

```
DELETE  https://{hostname|IPv4}/api/site/{siteId}
```

Note: To get a {site Id} value based on its name, follow the information in “Getting a {siteId}” on page 240.

Parameters

None.

Data

None.

Example: Delete a site

Assume that you have a site, Singapore, and you want to delete it. You get the `{siteId}`: 3101.

A Python snippet that is similar to the following example can be used to delete the site:

```
site_id = "3101"    # Singapore

requests.delete('https://' + spp_ipv4 + '/api/site/' + site_id,
    headers={...}, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 204 (No Content) and that you no longer see the site Singapore.

LDAP/SMTP Servers

With IBM Spectrum Protect Plus, you can add an individual user or add multiple users by specifying a Lightweight Directory Access Protocol (LDAP) server with LDAP groups and user accounts. You can add up to one LDAP server. The LDAP directory service on it must be either Microsoft Active Directory or OpenLDAP. For general information regarding LDAP, see the following standards document:

- Lightweight Directory Access Protocol (LDAP): The Protocol, IETF RFC 4511

For task instructions about using an LDAP server, see the following sections:

- "Adding users that are based on an LDAP group" on page 196
- "Modifying settings of users" on page 198

The Simple Mail Transfer Protocol (SMTP) is used for reporting features. When you add an SMTP server to IBM Spectrum Protect Plus, reports can be sent to specific addresses via email. For general information about SMTP, see the following standards document:

- Simple Mail Transfer Protocol, IETF RFC 5321

For task instructions about using an SMTP server, see the following section:

- "Scheduling a report" on page 161

Getting LDAP server information

You can get information about a registered LDAP server. The LDAP server list will display a maximum of one item because IBM Spectrum Protect Plus supports the registration of only one LDAP server.

Method and URI

To get information about an LDAP server, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/ldap
```

Parameters

None.

Data

None.

Example: Get information about an LDAP server

Assume that you registered an LDAP server with IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to request information about the LDAP server:

```
requests.get('https://' + spp_ipv4 + '/api/ldap',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "ldapServers": [
        {
            "links": {
                "self": {
                    "rel": "self",
                    "href": "https://10.0.0.111/api/ldap/1012",
                    ...
                },
            ...
            }
            "tenantId": 1000,
            "hostAddress": "10.0.0.111",
            "name": "10.0.0.111",
            "comment": null,
            "user": {
                "href": "https://10.0.0.100/api/identity/user/2148"
            },
            "sslConnection": false,
            "portNumber": 389,
            "baseDN": "DC=BLUEMACHINES,DC=com",
            "userRDN": "OU=SALES",
            "groupRDN": "OU=BM,OU=Security Groups",
            "userFilter": "sAMAccountName={0}",
            "rbacPath": "ldap:0/ldap:1012",
            "resourceType": "ldap",
            "id": "1012"
        }
    ]
}
```

Adding an LDAP server

You can add an LDAP server.

Method and URI

To add an LDAP server, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/ngp/ldap
```

Parameters

None.

Data

Data 1: `hostAddress`

IPv4 address of the LDAP server.

- **Example value:** 10.0.0.111
- **Type:** String. Required. Available in the web user interface.

Data 2: `portNumber`

Port number to connect to the LDAP server. The typical default port is 389 for non-SSL connections or 636 for SSL connections.

- **Example value:** 389
- **Type:** Integer. Required. Available in the web user interface.

Data 3: `sslConnection`

Indication of whether SSL is used to connect with the LDAP server.

- **Example value:** false (default)
- **Type:** Boolean. Required. Available in the web user interface.

Data 4: `useExistingUser`

Indication of whether an existing identity is used.

- **Example value:** true
- **Type:** Boolean. Required. Available in the web user interface.

Data 5: `existingUser`

The `{identityUserHref}` of the identity.

Tip: To get an {identityKeyHref} value based on the name of the identity, follow the information in "Getting an {identityKeyHref}" on page 233.

- **Example value:** https://10.0.0.100/api/identity/user/2148
- **Type:** String. Required. Available in the web user interface.

Data 6: `username`

The `{identityUserHref}` of the identity. This value should be the same as the `existingUser` value.

- **Example value:** https://10.0.0.100/api/identity/user/2148
- **Type:** String. Required. Available in the web user interface.

Data 7: `baseDN`

The base distinguished name (DN) of the LDAP object.

- **Example value:** `DC=BLUEMACHINES, DC=com`
- **Type:** String. Required. Available in the web user interface.

Data 8: `userFilter`

Filter value to select specific users in the base DN.

- **Example value:** `sAMAccountName={0}`
- **Type:** String. Required. Available in the web user interface.

Data 9: `userRDN`

Relative distinguished names of users, separated by commas.

- **Example value:** `cn=Users`
- **Type:** String. Required. Available in the web user interface.

Data 10: `groupRDN`

Relative distinguished names of groups, separated by commas.

- **Example value:** `OU=BM,OU=Security Groups`
- **Type:** String. Required. Available in the web user interface.

Example: Add an LDAP server

Assume that you want to add the following LDAP server with an Active Directory database to IBM Spectrum Protect Plus:

- **Host address:** 10.0.0.111
- **Port:** 389
- **SSL:** Disable
- **Credentials:** BLUEMACHINESSALES (`{identityUserHref}` https://10.0.0.100/api/identity/user/2148)

- **Base DN:**
  - **Domain component 1:** BLUEMACHINES
  - **Domain component 2:** com
- **User filter:** `sAMAccountName={0}`
  - Active Directory authentication with the Windows user naming attribute, `sAMAccountName`.
- **User RDN:**
  - **Organization unit:** SALES
- **Group RDN:**
  - **Organization unit 1:** BM
  - **Organization unit 2:** Security Groups

A Python snippet that is similar to the following example can be used to add the LDAP server:

```
_data = f'''{{
    "hostAddress":    "10.0.0.111",
    "portNumber":      389,
    "sslConnection":   false,
    "useExistingUser": true,
    "existingUser":    "https://10.0.0.100/api/identity/user/2148",
    "userame":         "https://10.0.0.100/api/identity/user/2148",
    "baseDN":          "DC=BLUEMACHINES,DC=com",
    "userFilter":      "sAMAccountName={{0}}",
    "userRDN":         "OU=SALES",
    "groupRDN":        "OU=BM,OU=Security Groups"
    }}'''

requests.post('https://' + spp_ipv4 + '/ngp/ldap',
    headers={...}, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created). Review the response to ensure that the LDAP server was added.

```
{
    "statusCode": 201,
    "response": {
        "links": {...},
        "tenantId": 1000,
        "hostAddress": "10.0.0.111",
        "name": "10.0.0.111",
        "comment": null,
        "user": {
            "href": "https://10.0.0.100/api/identity/user/2148"
        },
        "sslConnection": false,
        "portNumber": 389,
        "baseDN": "DC=BLUEMACHINES,DC=com",
        "userRDN": "OU=SALES",
        "groupRDN": "OU=BM,OU=Security Groups",
        "userFilter": "sAMAccountName={0}",
        "rbacPath": "ldap:0/ldap:1012",
        "resourceType": "ldap",
        "id": "1012"
    }
}
```

Editing the configuration of an LDAP server

You can edit an LDAP server configuration.

Method and URI

An LDAP server must be specified by using an `{ldapId}`. To update the configuration of an LDAP server, use a PUT method and a URI:

```
PUT     https://{hostname|IPv4}/ngp/ldap/{ldapId}
```

Tip: To get an {ldapId}, follow the information in “Getting an {ldapId} ” on page 235.

Parameters

None.

Data

You can use the data key-value pairs that are described in

“Adding an LDAP server” on page 132.

Example: Edit the configuration of an LDAP server

Assume that you want to edit the configuration of an LDAP server with the following `{ldapId}`: 1012. You want to assign the following values to this LDAP server:

- **Port:** 636 (LDAP over SSL)
- **SSL:** Enabled

A Python snippet that is similar to the following example can be used to update the configuration of the LDAP server:

```
ldap_id = "1012"

_data = f'''{{
    "portNumber":    636,
    "sslConnection": true
}}'''

requests.put('https://' + spp_ipv4 + '/ngp/ldap/' + ldap_id,
    headers={...}, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 200 (OK) and that the LDAP server settings are updated. The response body is similar to the JSON object for Sales_Americas, as shown in “Getting LDAP server information” on page 131.

Unregistering an LDAP server

You can unregister an LDAP server from IBM Spectrum Protect Plus.

Method and URI

To delete an LDAP server with `{ldapId}`, the request's method and URI are similar to the following example:

```
DELETE  https://{hostname|IPv4}/api/ldap/{ldapId}
```

Tip: To get an {ldapId} value based on its name, follow the information in “Getting an {ldapId} ” on page 235.

Parameters

None.

Data

None.

Example: Unregister an LDAP server

Assume that you have an LDAP server, 10.0.0.111, and you want to delete it. You get the {ldapId}: 1011.

A Python snippet that is similar to the following example can be used to unregister the LDAP server:

```
ldap_id = "1011"     # 10.0.0.111

requests.delete('https://' + spp_ipv4 + '/api/ldap/' + ldap_id,
    headers={...}, params="", data="", verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 204 (No Content) and that you no longer see the LDAP server 10.0.0.111. The response body is empty.

Getting SMTP server information

You can get information about a registered SMTP server. For each IBM Spectrum Protect Plus server, you can register only one SMTP server.

Method and URI

To get information about an SMTP server, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/smtp/{smtpId}
```

Tip: To get an {smtpId} value based on its name, follow the information in “Getting an {smtpId}” on page 241.

Parameters

None.

Data

None.

Example: Get information about an SMTP server

Assume that you registered an SMTP server with IBM Spectrum Protect Plus ({smtpId} 1002).

A Python snippet that is similar to the following example can be used to request information about this SMTP server:

```
smtp_id = "1002"

requests.get('https://' + spp_ipv4 + '/api/smtp/' + smtp_id,
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
```

```
        "name": "10.0.0.121",
        "hostAddress": "10.0.0.121"
        "rbacPath": "smtp:0/smtp:1002",
        "user": "sarah-smtp",
        "comment": "",
        "portNumber": 25,
        "timeout": 10000,
        "fromAddress": "smtp@bluemachines.com.invalid",
        "subjectPrefix": "IBM Spectrum Protect
Plus:",
        "id": "1002"
    }
```

Adding an SMTP server

You can add an SMTP server to IBM Spectrum Protect Plus.

Method and URI

To add an SMTP server, use a POST method with a URI:

```
    POST    https://{hostname|IPv4}/ngp/smtp
```

Parameters

None.

Data

Data 1: `hostAddress`

The IP address of the host, or the path and hostname of the SMTP server.

- **Example value:** 10.0.0.121
- **Type:** String. Required. Available in the web user interface.

Data 2: `portNumber`

The communications port of the server that you are adding. The default port number is 25 for non-SSL connections or 443 for SSL connections.

- **Example value:** 25
- **Type:** Integer. Required. Available in the web user interface.

Data 3: `username`

The name that is used to access the SMTP server.

- **Example value:** sarah-smtp
- **Type:** String. Available in the web user interface.

Data 4: `password`

The name that is used to access the SMTP server.

- **Example value:** ?^y{+~k9R8S*S4FW
- **Type:** String. Required, if `username` has a value. Available in the web user interface.

The password that is associated with the username.

Data 5: `timeout`

The email timeout value in milliseconds.

- **Example value:** 10000
- **Type:** Integer. Available in the web user interface.

Data 6: `fromAddress`

The address that is associated with email communications from IBM Spectrum Protect Plus.

- **Example value:** smtp@bluemachines.com.invalid
- **Type:** String. Available in the web user interface.

Data 7: `subjectPrefix`

The prefix for email subject lines sent from IBM Spectrum Protect Plus.

- **Example value:** IBM Spectrum Protect Plus:
- **Type:** String. Available in the web user interface.

Example: Add an SMTP server to IBM Spectrum Protect Plus

Assume that you want to add the following SMTP server to IBM Spectrum Protect Plus:

- **Host address:** 10.0.0.121
- **Port:** 25 (SMTP without SSL encryption)
- **Username:** sarah-smtp
- **Password:** ?^y{+~k9R8S⋆S4FW
- **Timeout:** 10,000 milliseconds (10 seconds)
- **From address:** smtp@bluemachines.com.invalid
- **subjectPrefix:** IBM Spectrum Protect Plus:

A Python snippet that is similar to the following example can be used to add the SMTP server:

```
   _data = f'''{{
       "hostAddress":   "10.0.0.121",
       "portNumber":    25,
       "username":      "sarah-smtp",
       "password":      "?^y{{+~k9R8S⋆S4FW",
       "timeout":       10000,
       "fromAddress":   "smtp@bluemachines.com.invalid",
       "subjectPrefix": "IBM Spectrum Protect
Plus :"
   }}'''

   requests.post('https://' + spp_ipv4 + '/ngp/smtp',
       headers={...}, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created). Review the response to ensure that the SMTP server 10.0.0.121 was added. The response body is similar to the JSON object that is shown in “Getting SMTP server information” on page 136.

Editing the configuration of an SMTP server

You can edit an SMTP server configuration.

Method and URI

To edit the configuration of the SMTP server, use a PUT method with a URI:

```
   PUT     https://{hostname|IPv4}/ngp/smtp/{smtpId}
```

Tip: To get an {smtpId}, follow the information in “Getting an {smtpId}” on page 241.

Parameters

None.

Data

You can use the data key-value pairs that are listed in “Adding an SMTP server” on page 137.

Example: Edit the configuration of an SMTP server

Assume that you want to edit the configuration of an SMTP server with the following {smtpId} value: 1002. You want to assign the following values to this SMTP server:

- **Port:** 443 (HTTP with SSL encryption)
- **From address** smtp.dallas@bluemachines.com.invalid

A Python snippet that is similar to the following example can be used to edit the configuration of the SMTP server:

```
   smtp_id = "1002"

   _data = f'''{{
       "hostAddress":   "10.0.0.121",
       "portNumber":    443,
       "username":      "sarah-smtp",
       "password":      "?^y{{+~k9R8S*S4FW",
       "timeout":       10000,
       "fromAddress":   "smtp.dallas@bluemachines.com.invalid",
       "subjectPrefix": "IBM Spectrum Protect
Plus:"
   }}'''

   requests.put('https://' + spp_ipv4 + '/ngp/smtp/' + smtp_id,
       headers={...}, params="", data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 200 (OK) and that the SMTP server settings are updated.

Deleting an SMTP server

To unregister an SMTP server from IBM Spectrum Protect Plus, you must delete the SMTP server.

Method and URI

To unregister an SMTP server, use a DELETE method and a URI:

```
DELETE  https://{hostname|IPv4}/api/smtp/{smtpId}
```

Note: To get an {smtpId} value based on the SMTP server name, follow the information in “Getting an {smtpId}” on page 241.

Parameters

None.

Data

None.

Example: Unregister an SMTP server

Assume that you have an SMTP server, 10.0.0.121, and you want to unregister it. You get the `{smtpId}`: 1002.

A Python snippet that is similar to the following example can be used to unregister the SMTP server:

```
smtp_id = "1002"

requests.delete('https://' + spp_ipv4 + '/api/smtp/' + smtp_id,
    headers={...}, params="", data="", verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 204 (No Content) and that you no longer see the SMTP server 10.0.0.121. The response body is empty.

Keys and Certificates

In the Keys and Certificates section of the web user interface, you can manage access keys, SSH private keys, and certificates that are registered with IBM Spectrum Protect Plus. The following tables summarize where you can manage authentication methods for each type of product.

Virtualized systems:

Product	Access keys	SSH keys	Certificates	Identity	User menu
VMware vCenter				x	
Hyper-V				x	
Amazon EC2				x	
Linux		x		x	
Windows				x	

Application servers:

Product	Access keys	SSH keys	Certificates	Identity	User menu
Oracle Database		x		x	
SQL Server				x	
Db2		x		x	
MongoDB		x		x	
Exchange Server				x	
Microsoft 365				x	

Object storage:

Product	Access keys	SSH keys	Certificates	Identity	User menu
Amazon S3	x				
IBM Cloud Object Storage	x			x	
Azure Blob Storage	x				
S3 compatible object storage	x			x	

Repository servers:

Product	Access keys	SSH keys	Certificates	Identity	User menu
IBM Spectrum Protect Plus	x		x		

IBM Spectrum Protect Plus:

Product	Access keys	SSH keys	Certificates	Identity	User menu
IBM Spectrum Protect Plus					x
Individual users				x	
LDAP server				x	
SMTP server				x	

To manage usernames and passwords, follow the instructions in “Identity” on page 214. To establish a secure connection between IBM Spectrum Protect Plus and a client, for example, your workstation, you must upload an SSL certificate file by clicking the user menu and clicking **Manage SSL Certificates**. For details, see the IBM Spectrum Protect Plus User's and Installation Guide.

Getting information about access keys and SSH keys

You can get a list of access keys and SSH keys with information about each key. Access keys and SSH keys share the same URI path. You can use the `filter` parameter to get the keys of either type.

Method and URI

To get information about all access keys and SSH keys, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/identity/key
```

To get information about a specific access key or an SSH key, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/identity/key/{identityKeyId}
```

Tip: To get an {identityKeyId} value based on its name, follow the information in “Getting an {identityKeyId}” on page 233.

Parameters

Parameter 1: `filter`

Specify criteria to get access keys or SSH keys.

- **Values:** To get only access keys, use the following value:

```
[
    {
        "property": "keytype",
        "value":    ["exch_key"],
        "op":       "NOT IN"
    }
]
```

To get only SSH keys, use the following value:

```
[
    {
        "property": "keytype",
        "value":    "ssh_private_key",
        "op":       "="
    }
]
```

- **Type:** List. Available in the web user interface.

Parameter 1.1: `filter` > `property`

Specify a property to apply to the filter.

- **Value:** `keytype`
- **Type:** List. Available in the web user interface.

Parameter 1.2: `filter` > `value`

Specify the property value that you want to apply to the filter.

- **Type:** List or system string. Available in the web user interface.

Data

None.

Example 1: Get information about all access keys

Assume that you added access keys to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to request information about the access keys:

```
_params = {
    "filter": str([{
        "property": "keytype",
        "value":    ["exch_key"],
        "op":       "NOT IN"
    }])
}

requests.get('https://' + spp_ipv4 + '/api/identity/key',
    params=_params, headers={...}, verify=...)
```

The response body for the Python snippet displays a list of all access keys and all SSH keys. The HTTP status of this response is 200 (OK).

```
{
    "links": {...},
    "total": 3,
    "keys": [
        {
            "links": {...},
            "resourceType": "key",
            "id": "1002",
            "name": "BlueMachines Amazon EC2 Key1",
            "keytype": "iam_key",
```

```
            "keyalgorithm": null,
            "length": 0,
            "user": null,
            "comment": null,
            "fingerprint": null,
            "publickey": null,
            "access": "G9493LAPUDNDPL9J38SW",
            "secret": null,
            "privatekey": "",
            "tenant": null,
            "tenantID": null,
            "clientID": null,
            "clientSecret": null,
            "adminUser": null,
            "adminUserPass": null,
            "rbacPath": "root:0/identity:0/identitytype:key/identitykey:1002"
        },
        {
            ...
            "id": "1003",
            "name": "BlueMachines IBM COS Dallas Key",
            ...
        },
        {...}
    ]
}
```

Example 2: Get information about all SSH keys

Assume that you added SSH keys to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to request information about the SSH keys:

```
_params = {
    "filter": str([{
        "property": "keytype",
        "value":    "ssh_private_key",
        "op":       "="
    }])
}

requests.get('https://' + spp_ipv4 + '/api/identity/key',
    params=_params, headers={...}, verify=...)
```

The response body for the Python snippet displays a list of all access keys and all SSH keys. The HTTP status of this response is 200 (OK).

```
{
    "links": {...},
    "total": 2,
    "keys": [
        {
            "links": {...},
            "resourceType": "key",
            "id": "1006",
            "name": "Oracle - Dallas 1 SSH",
            "keytype": "ssh_private_key",
            "keyalgorithm": "RSA",
            "length": 1024,
            "user": "sarah-oracle1",
            "comment": null,
            "fingerprint": "5f:96:57:e2:43:52:16:25:c2:27:25:e7:39:9f:05:b7",
            "publickey": "ssh-rsa 22EYOeQkIV0OlvepsVdy2NoktYdwZhfR9JFMNkUj3IXd3p
   ...
   0pvuGdiXPlEV/XxE3yqRDGqwiVQYMSOZd9wgfjOFMbq9DkTZYpfivZXjzFWoJVKNYSx ",
            "access": null,
            "secret": "",
            "privatekey": null,
            "tenant": null,
            "tenantID": null,
            "clientID": null,
            "clientSecret": null,
            "adminUser": null,
            "adminUserPass": null,
            "rbacPath": "root:0/identity:0/identitytype:key/identitykey:1006"
        },
```

```
        {
            ...
            "id": "1007",
            "name": "support-vm SSH",
            ...
        }
    ]
}
```

Example 3: Get information about a specific access key

In this example, a Python snippet is used to get information about a specific access key: BlueMachines IBM COS Dallas Key ({identityKeyId} 1003).

```
identity_key_id = "1003"

requests.get('https://' + spp_ipv4 + '/api/identity/key/' + identity_key_id,
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "resourceType": "key",
    "id": "1003",
    "name": "BlueMachines IBM COS Dallas Key",
    "keytype": "iam_key",
    "keyalgorithm": null,
    "length": 0,
    "user": null,
    "comment": null,
    "fingerprint": null,
    "publickey": null,
    "access": "COJf710BhRzrCvARh7gH",
    "secret": null,
    "privatekey": "",
    "tenant": null,
    "tenantID": null,
    "clientID": null,
    "clientSecret": null,
    "adminUser": null,
    "adminUserPass": null,
    "rbacPath": "root:0/identity:0/identitytype:key/identitykey:1003"
}
```

Adding an access key

You can add an access key.

Method and URI

To add an access key, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/api/identity/key
```

Parameters

None.

Data

Data 1: name

The name of the access key.

- **Example value:** BlueMachines Amazon S3 California Key

- **Type:** String. Required. Available in the web user interface.

Data 2: `keytype`

The type of the access key.

- **Value:** `iam_key`
- **Type:** System string. Required.

Data 3: `access`

An access key string.

- **Example value:** `G9493LAPUDNDPL9J38SW`
- **Type:** String. Available in the web user interface.

Data 4: `secret`

A secret key of the access key.

- **Example value:** `rNzSHk2gBfjf9cECGp+cjrhEdhsZmxwHnRckk8BP`
- **Type:** String. Available in the web user interface.

Example: Add an access key

Assume that you want to add an access key with the following properties:

- Name: BlueMachines S3 Compatible Object Storage Key
- Access key: `PJNG6J5FHENHWZ66K77X`
- Secret key: `CFBF8QcKtMSMYTE++mX3pUWSwmchNBgg5afbnw5T`

A Python snippet that is similar to the following example can be used to send a request to IBM Spectrum Protect Plus to add the access key:

```
_data = f'''{{
    "name":    "BlueMachines S3 Compatible Object Storage Key",
    "keytype": "iam_key",
    "access":  "PJNG6J5FHENHWZ66K77X",
    "secret":  "CFBF8QcKtMSMYTE++mX3pUWSwmchNBgg5afbnw5T"
}}'''

requests.post('https://' + spp_ipv4 + '/api/identity/key',
    headers={...}, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 201 (Created) and that the access key was created. The response body is similar to the JSON object for an access key, as shown in

“Getting information about access keys and SSH keys” on page 141.

Adding an SSH key

Add an SSH key.

Tip: To create an SSH key on the target Linux server, follow the instructions in the IBM Spectrum Protect Plus User's and Installation Guide.

Method and URI

To add an SSH key, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/api/identity/key
```

Parameters

None.

Data

Data 1: `name`

The name of the SSH key.

- **Example value:** BlueMachines Amazon S3 California Key
- **Type:** String. Required. Available in the web user interface.

Data 2: `user`

A user name of the target Linux server for which you want to establish an SSL connection.

- **Example value:** `sarah-oracle1`
- **Type:** String. Required. Available in the web user interface.

Data 3: `keytype`

The type of key.

- **Value:** `ssh_private_key`
- **Type:** System string. Required. Available in the web user interface.

Data 4: `privatekey`

The private SSH key of the target Linux server. Replace new lines with the escape sequence characters:

```
\n
```

- **Example value:**

```
-----BEGIN RSA PRIVATE KEY-----\nProc-Type: 4,ENCRYPTED\nDEK-Info: AES-1
   28-CBC,08295638...\n\nUbdH2oDSWTCkWrxWsLaDvEvV77O4vZhxDkcE6plRYyp8d4aCQ
   ...
   wS1Ff2nxnW/7A9i9yKRGvFIiaotFIoa43ZLA8ivn\n-----END RSA PRIVATE KEY-----
```

- **Type:** String. Required. Available in the web user interface.

Data 5: `passphrase`

The optional passphrase for the SSH key of the target Linux server. It may be different from the user password of the Linux server.

- **Example value:** `E^vBQ:9!Y)w<pNw~`
- **Type:** String. Available in the web user interface.

Example: Create an SSH key

Assume that you want to establish a secure connection between IBM Spectrum Protect Plus and a Linux server that hosts an Oracle Database. You log in to the Linux server and generate an SSH public key and private key:

```
[sarah@workstation ~ ] ssh sarah-oracle1@10.4.4.1  # oracle-dallas1
sarah-oracle1@10.4.4.1's password: A'mK'"~RPtY=*k(h
...
```

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sarah-oracle1/.ssh/id_rsa):
Created directory '/home/sarah-oracle1/.ssh'.
```

```
Enter passphrase (empty for no passphrase): E^vBQ:9!Y)w<pNw~
...
Your identification has been saved in /home/sarah-oracle1/.ssh/id_rsa.
Your public key has been saved in /home/sarah-oracle1/.ssh/id_rsa.pub.
...
The key's randomart image is:
+---[RSA 2048]----+
|     +.o=o  o    |
...
+----[SHA256]-----+
```

```
$ cat /home/sarah-oracle1/.ssh/id_rsa.pub
ssh-rsa 22EYOeQkIV0Olveps...
```

```
$ cat /home/sarah-oracle1/.ssh/id_rsa.pub \
  >> /home/sarah-oracle1/.ssh/authorized_keys
```

```
$ chmod 600 \
  /home/sarah-oracle1/.ssh/authorized_keys
```

```
$ sudo cp /etc/ssh/sshd_config{,.bak}
```

```
$ sudo vi /etc/ssh/sshd_config    # Enable: PubkeyAuthentication yes
```

```
$ systemctl restart sshd
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to manage system services or units.
Multiple identities can be used for authentication:
 1.  oracle
 2.  grid
 3.  sarah-oracle1
Choose identity to authenticate as (1-3): 3
Password: A'mK'"~RPtY=*k(h
==== AUTHENTICATION COMPLETE ===
```

```
$ cat /home/sarah-oracle1/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,4E9A0765C9067B52F06549D610C6591A

xbeIIcosQ5TsuqXirKJFlgOz1ryQus7RrapZcBgSuZ8PdVXvtvYDBA2qRo79G4OY
...
Pnidu8/8iFy7uoNe9F5J6h40g6S1vI58Ub/e9fbgIzd2NO5MhYkc5c0a8I1rhUq1
-----END RSA PRIVATE KEY-----
$
```

The SSH key that you have generated on the Linux server has the following properties:

- **Name:** Oracle - Dallas 1 SSH
- **User:** `sarah-oracle1`
- **Private key:** See the Python snippet.
- **Passphrase:** `E^vBQ:9!Y)w<pNw~`

A Python snippet that is similar to the following example can be used to send a request to IBM Spectrum Protect Plus to add the new SSH key:

```
privatekey_original = f'''-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,4E9A0765C9067B52F06549D610C6591A

xbeIIcosQ5TsuqXirKJFlgOz1ryQus7RrapZcBgSuZ8PdVXvtvYDBA2qRo79G4OY
...
```

```
Pnidu8/8iFy7uoNe9F5J6h40g6S1vI58Ub/e9fbgIzd2NO5MhYkc5c0a8I1rhUq1
-----END RSA PRIVATE KEY-----'''


privatekey = privatekey_original.replace('\n', '\\n')  # Replace new lines with \n

_data = f'''{{
    "name":       "Oracle - Dallas 1 SSH",
    "user":       "sarah-oracle1",
    "keytype":    "ssh_private_key",
    "privatekey": "{privatekey}",
    "passphrase": "E^vBQ:9!Y)w<pNw~"
}}'''

requests.post('https://' + spp_ipv4 + '/api/identity/key',
    headers={...}, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 200 (OK) and that the SSH key settings are updated. This request is similar to the one that you used to create the SSH key; compare the HTTP request method, the URI, the `data` values, and the HTTP status code in the response. The response body is similar to the JSON object for the original SSH key, as shown in "Getting information about access keys and SSH keys" on page 141.

Editing the configuration of an access key or an SSH key

You can edit the configuration of an access key or an SSH key.

Method and URI

To edit the configuration of an access key, use the PUT method with a URI:

```
PUT     https://{hostname|IPv4}/api/identity/{identityKeyId}
```

Note: To get an {identityKeyId}, follow the information in "Getting an {identityKeyId}" on page 233.

Parameters

None.

Data

You can use the data key-value pairs that are described in "Adding an access key" on page 144 and "Adding an SSH key" on page 145.

Example: Edit the configuration of an access key

Assume that you want to edit the access key BlueMachines IBM Amazon S3 US-East1 Key (`{identityKeyId}` 1005). You want to assign the following values to this access key:

- Name: BlueMachines IBM COS Singapore Key
- Access key: `FD82Hf2w37tqq2sX9Uye`
- Secret key: `fVBvcjgJvbjvuWBPU7uVzTJc47CNNJtQmnJNJted`

Run the following command:

```
identity_key_id = "1005"

_data = f'''{{
    "name":   "BlueMachines IBM COS Singapore Key,
    "access": "FD82Hf2w37tqq2sX9Uye"
    "secret": "fVBvcjgJvbjvuWBPU7uVzTJc47CNNJtQmnJNJted"
}}'''

requests.put('https://' + spp_ipv4 + '/api/identity/key/' + identity_key_id,
    headers={...}, params="", data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 200 (OK) and that the access key settings are updated. This request is similar to the one that you used to create the access key; compare the HTTP request method, the URI, the `data` values, and the HTTP status code in the response. The response body is similar to the JSON object for the original access key, as shown in "Getting information about access keys and SSH keys" on page 141.

Deleting an access key or an SSH key

You can delete an access key or an SSH key.

Method and URI

To delete an access key or an SSH key with `{identityKeyId}`, use a DELETE method and a URI:

```
DELETE  https://{hostname|IPv4}/api/identity/key/{identityKeyId}
```

Tip: To get an {identityKeyId} value based on its name, follow the information in "Getting an {identityKeyId}" on page 233.

Parameters

None.

Data

None.

Example: Delete an access key

Assume that you want to delete the access key, BlueMachines Amazon EC2 Key1 (`{identityKeyId}` 1002). A Python snippet that is similar to the following example can be used to delete the access key:

```
identity_key_id = "1002"    # BlueMachines Amazon EC2 Key1

requests.delete('https://' + spp_ipv4 + '/api/identity/key/' + identity_key_id,
    headers={...}, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 204 (No Content) and that you no longer see the access key. You can delete an SSH key in a similar way.

Getting certificate information

You can get certificate information.

Method and URI

To get information about all certificates, use a GET method with a URI:

```
GET     https://{hostname|IPv4}/api/security/certificate
```

To get information about a specific certificate, use a GET method with a URI:

```
GET     https://{hostname|IPv4}/api/security/certificate/{certificateId}
```

Note: To get a {certificateId} value based on its name, follow the information in "Getting a {certificateId}" on page 239.

Parameters

None.

Data

None.

Example 1: Get information about all certificates

Assume that you want to get information about all certificates.

A Python snippet that is similar to the following example can be used to get information about all certificates:

```
requests.get('https://' + spp_ipv4 + '/api/security/certificate',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "total": 3,
    "certificates": [
        {
            "links": {...},
            "name": "BlueMachines - Cert IBM Spectrum Protect",
            "displayName": null,
            "id": 1286,
            "storeId": "1286",
            "type": "sp",
            "lastUpdated": 1580536800000,
            "comment": null,
            "rbacPath": "root:0/certificate:0/certificate.type:sp/certificate:1286"
        },
        {
            "links": {...},
            "name": "BlueMachines - Cert IBM Cloud Object Storage",
            "displayName": null,
            "id": 1287,
            "storeId": "1287",
            "type": "cos",
            "lastUpdated": 1577858400000,
            "comment": null,
            "rbacPath": "root:0/certificate:0/certificate.type:cos/certificate:1287"
        },
        {
            "links": {...},
            "name": "BlueMachines - Cert S3 Compatible Object Storage",
            "displayName": null,
            "id": 1288,
            "storeId": "1288",
            "type": "aws",
            "lastUpdated": 1583042400000,
            "comment": null,
            "rbacPath": "root:0/certificate:0/certificate.type:aws/certificate:1288"
        }
    ]
}
```

Example 2: Get information about a specific certificate

Assume that you want to get information about the following certificate: BlueMachines - Cert IBM Spectrum Protect (`{certificateId}` 1286). A Python snippet that is similar to the following example can be used to request the information about this certificate:

```
certificate_id = "1286"

requests.get('https://' + spp_ipv4 + '/api/security/certificate/'
```

```
        + certificate_id,
        headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "name": "BlueMachines - Cert IBM Spectrum Protect",
    "displayName": null,
    "id": 1286,
    "storeId": "1286",
    "type": "sp",
    "lastUpdated": 1587187415947,
    "comment": null,
    "rbacPath": "root:0/certificate:0/certificate.type:sp/certificate:1286"
}
```

Adding a certificate

In the REST API, you can add a new certificate by sending the certificate string in plain text format.

You cannot edit configurations of certificates that you added. To update a certificate, delete it and add a new certificate.

Method and URI

To add a certificate, use a POST method and a URI:

```
POST    https://{hostname|IPv4}/api/security/certificate
```

Parameters

None.

Data

Data 1: `crt`

The certificate in a text format. Replace new lines with the escape sequence characters:

```
\n
```

- **Example value:** `-----BEGIN CERTIFICATE-----\nj1w5eFM...+Vrr=\n-----END CERTIFICATE-----`
- **Type:** String. Required. Available in the web user interface.

Data 2: `name`

A meaningful name for the certificate.

- **Example value:** BlueMachines Cert IBM Spectrum Protect
- **Type:** String. Required. Available in the web user interface.

Data 3: `type`

The type of certificate. Use one of the following system values:

Type	Value
S3 compatible object storage	aws
IBM Spectrum Protect	sp
IBM Cloud Object Storage	cos

- **Type:** System string. Required. Available in the web user interface.

Restriction: IBM Spectrum Protect Plus does not support certificate management for the original Amazon S3.

Example: Add a certificate

Assume that you want to add the following certificate text to IBM Spectrum Protect Plus:

- Certificate value: See the Python snippet.
- Name: BlueMachines - Cert IBM Spectrum Protect
- Type: IBM Spectrum Protect

A Python snippet that is similar to the following example can be used to add the certificate:

```
cert_original = f'''-----BEGIN CERTIFICATE-----
j1w5eFMwPNhJSfl4zAau4GzOUuPyZea2OP5Arvsg00BKbr2dLwZ6EyON1Ux4ab0FMl2PHvD5ERthstuH
XpTKcSb66fI2hmyegTbV3EQCPR+GJJcCbSpBngu2u4onyLlr4V7gkvmVPj9yvgS64Co4BS3kwHCOtVen
...
vfJdE2s4Zc31HvMSKOBy37YTtNADXqEIle+Vrr=
-----END CERTIFICATE-----'''

cert_string = cert_original.replace('\n', '\\n')  # Replace new lines with \n

data = f'''{{
    "cert_value": {cert_string},
    "name": "BlueMachines - Cert IBM Spectrum Protect"
    "type": "sp"
}}'''

requests.post('https://' + spp_ipv4 + '/api/security/certificate',
    headers={...}, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created). The response body is similar to the JSON object for the certificate, "BlueMachines - Cert IBM Spectrum Protect", as shown in “Getting certificate information” on page 149.

Deleting a certificate

You can delete a certificate from IBM Spectrum Protect Plus.

Method and URI

To delete a certificate, use a DELETE method with a URI:

```
DELETE  https://{hostname|IPv4}/api/security/certificate/{certificateId}
```

Note: To get the {certificateId} based on the certificate name, follow the instructions in “Getting a {certificateId}” on page 239.

Parameters

None.

Data

None.

Example: Delete a certificate

Assume that you have a certificate, BlueMachines - Cert S3 Compatible, with a `{certificateId}` value of 1288, and you want to delete the certificate. Run the following command:

```
certificate_id = "1288"

requests.delete('https://' + spp_ipv4 + '/api/security/certificate/'
    + certificate_id,
    headers={...}, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 204 (No Content) and that you no longer see this certificate.

Global Preferences Overview

Global Preferences are settings that can be modified in the IBM Spectrum Protect Plus UI to fine tune settings to match your environment. These settings range from the ability to tweak the amount of time before the user interface of IBM Spectrum Protect Plus logs the user out in minutes, the application inventory SSH timeout in hours, the access log retention in days, the ability to set the VADP proxies in the environment to use only the HotAdd transport mode, and to set the minimum password length for IBM Spectrum Protect Plus, among several others. These settings may be retrieved and modified through the IBM Spectrum Protect Plus REST API.

Important: Global Preference settings should only be changed at the direction of IBM Support.

You can complete operations with the REST API, as described in the following sections.

Getting information about Global Preferences

You can get information about Global Preferences.

Method and URI

To get information about all Global Preferences, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/preference
```

To get information about a specific Global Preference, use a GET method and a URI with the `{globalpreferenceId}`:

```
GET     https://{hostname|IPv4}/api/endeavour/preference/{globalpreferenceId}
```

Tip: To get a {globalpreferenceId} value based on the UI string name of the Global Preference, follow the information in “Getting an {globalpreferenceId}” on page 226.

Parameters

None.

Data

None.

Example 1: Get information about all Global Preferences

A Python snippet that is similar to the following example can be used to request a list of all Global Preferences and the associated information for each preference.

```
   requests.get('https://' + spp_ipv4 + '/api/endeavour/preference',
       headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
 {
    ...
    "preferences": [
        {
            "links": {
                "self": {
                    "rel": "self",
                    "href": "https://10.0.0.2/api/endeavour/preference/
pref.application.common.catalog.iaMappedLunsSkipped",
                    "hreflang": null,
                    "media": null,
                    "title": null,
                    "type": null,
                    "deprecation": null
                },
                "up": {
                    "rel": "up",
                    "href": "https://10.0.0.2/api/endeavour/preference",
                    "hreflang": null,
                    "media": null,
                    "title": null,
                    "type": null,
                    "deprecation": null
                }
            },
            "name": "Enable SQL Server and Oracle databases restored in test mode eligible for
backup",
            "nameKey": "pref.application.common.catalog.iaMappedLunsSkipped",
            "category": "Application",
            "categoryKey": "pref.category.application",
            "type": "boolean",
            "typeKey": "pref.type.boolean",
            "subtype": "pref.subtype.sql",
            "subtypeKey": "pref.subtype.sql",
            "version": null,
            "uiOption": null,
            "value": {
                "type": "boolean",
                "typeKey": "pref.type.boolean",
                "isArray": false,
                "defaultValue": false,
                "value": null,
                "minimum": null,
                "maximum": null,
                "unit": null,
                "unitKey": null,
                "format": null,
                "formatKey": null,
                "required": false,
                "readOnly": false
            },
            "id": "pref.application.common.catalog.iaMappedLunsSkipped"
        },
        {
            "links": {
                "self": {
```

```
                    "rel": "self",
                    "href": "https://10.0.0.2/api/endeavour/preference/
pref.application.common.inventorySshTimeout",
                    "hreflang": null,
                    "media": null,
                    "title": null,
                    "type": null,
                    "deprecation": null
                },
                "up": {
                    "rel": "up",
                    "href": "https://10.0.0.2/api/endeavour/preference",
                    "hreflang": null,
                    "media": null,
                    "title": null,
                    "type": null,
                    "deprecation": null
                }
            },
            "name": "Application inventory SSH timeout (hours)",
            "nameKey": "pref.application.common.inventorySshTimeout",
            "category": "Application",
            "categoryKey": "pref.category.application",
            "type": "integer",
            "typeKey": "pref.type.integer",
            "subtype": "common",
            "subtypeKey": "pref.subtype.common",
            "version": null,
            "uiOption": null,
            "value": {
                "type": "integer",
                "typeKey": "pref.type.integer",
                "isArray": false,
                "defaultValue": 12,
                "value": 27,
                "minimum": 1,
                "maximum": 2048,
                "unit": "hrs",
                "unitKey": "hrs",
                "format": null,
                "formatKey": null,
                "required": false,
                "readOnly": false
            },
            "id": "pref.application.common.inventorySshTimeout"
        },
        {
            "links": {
                "self": {
                    "rel": "self",
                    "href": "https://10.0.0.2/api/endeavour/preference/
pref.application.common.iscsi.maxvolumesize",
                    "hreflang": null,
                    "media": null,
                    "title": null,
                    "type": null,
                    "deprecation": null
                },
                "up": {
                    "rel": "up",
                    "href": "https://10.0.0.2/api/endeavour/preference",
                    "hreflang": null,
                    "media": null,
                    "title": null,
                    "type": null,
                    "deprecation": null
                }
            },
            "name": "Maximum volume size for backup target LUNs on Windows (TB)",
            "nameKey": "pref.application.common.iscsi.maxvolumesize",
            "category": "Application",
            "categoryKey": "pref.category.application",
            "type": "integer",
            "typeKey": "pref.type.integer",
            "subtype": "common",
            "subtypeKey": "pref.subtype.common",
            "version": null,
            "uiOption": null,
            "value": {
                "type": "integer",
                "typeKey": "pref.type.integer",
                "isArray": false,
```

```
                "defaultValue": 256,
                "value": null,
                "minimum": 0,
                "maximum": 256,
                "unit": "TB",
                "unitKey": "pref.unit.tb",
                "format": null,
                "formatKey": null,
                "required": false,
                "readOnly": false
            },
            "id": "pref.application.common.iscsi.maxvolumesize"
        },
    ...
    ]
}
```

Example 2: Get information about a specific Global Preference

Assume that you want information about a specific Global Preference with the {globalpreferenceId} of pref.application.common.inventorySshTimeout IBM Spectrum Protect Plus.

A Python snippet that is similar to the following example can be used to request information about this Global Preference:

```
   globalpreferenceId = "pref.application.common.inventorySshTimeout"

   requests.get('https://' + spp_ipv4 + '/api/endeavour/preference/' + globalpreferenceId,
       headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {
        "self": {
            "rel": "self",
            "href": "https://10.0.0.2/api/endeavour/preference/
pref.application.common.inventorySshTimeout",
            "hreflang": null,
            "media": null,
            "title": null,
            "type": null,
            "deprecation": null
        },
        "up": {
            "rel": "up",
            "href": "https://10.0.0.2/api/endeavour/preference",
            "hreflang": null,
            "media": null,
            "title": null,
            "type": null,
            "deprecation": null
        }
    },
    "name": "Application inventory SSH timeout (hours)",
    "nameKey": "pref.application.common.inventorySshTimeout",
    "category": "Application",
    "categoryKey": "pref.category.application",
    "type": "integer",
    "typeKey": "pref.type.integer",
    "subtype": "common",
    "subtypeKey": "pref.subtype.common",
    "version": null,
    "uiOption": null,
    "value": {
        "type": "integer",
        "typeKey": "pref.type.integer",
        "isArray": false,
        "defaultValue": 12,
        "value": 27,
        "minimum": 1,
        "maximum": 2048,
        "unit": "hrs",
        "unitKey": "hrs",
        "format": null,
```

```
        "formatKey": null,
        "required": false,
        "readOnly": false
    },
    "id": "pref.application.common.inventorySshTimeout"
}
```

Editing the value of a Global Preference

You can edit the value for a Global Preference. Global Preference settings should only be changed at the direction of IBM Support.

Method and URI

To update the value of a Global Preference, use a PUT method and a URI:

```
PUT     https://{hostname|IPv4}/api/endeavour/preference/{globalpreferenceId}
```

Tip: To get a {globalpreferenceId} value based on the UI string name of the Global Preference, follow the information in "Getting an {globalpreferenceId}" on page 226.

Parameters

None.

Data

Global Preference values are presented as key and value pairs. Update the setting for `value` for the Global Preference to be updated. Values are typically of the integer or Boolean type. For Global Preferences of the integer type, the minimum and maximum allows values are presented through the values for `minimum` and `maximum`.

Example: Edit the value of a Global Preference

Assume that you have been directed to change the `{globalpreferenceId}` of `pref.application.common.inventorySshTimeout` that has the UI string name of **Application inventory SSH timeout (hours)** to a value of 4. You want to update the **value** for this Global Preference:

- **Type:** Integer
- **Type Key:** pref.type.integer
- **Is Array:** False
- **Default Value:** 12
- **Value:** 4
- **Minimum:** 1
- **Maximum:** 2048
- **Unit:** HRS
- **Unit Key:** HRS
- **Format:** NULL
- **Format Key:** NULL
- **Required:** False
- **Read Only:** False

A Python snippet that is similar to the following example can be used to update the value for the Global Preference:

```
globalpreferenceId = "pref.application.common.inventorySshTimeout"

_data = f'''{
     "type": "integer",
     "typeKey": "pref.type.integer",
     "isArray": false,
     "defaultValue": 12,
     "value": 4,
     "minimum": 1,
     "maximum": 2048,
     "unit": "hrs",
     "unitKey": "hrs",
     "format": null,
     "formatKey": null,
     "required": false,
     "readOnly": false
 }'''

requests.put('https://' spp_ipv4 + '/api/endeavour/preference/' + globalpreferenceId,
    headers={...}, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 200 (OK) and that the value for the Global Preference is updated.

Chapter 6. Reports and Analytics

You can use the REST API to obtain reports and analyze system status.

Reports

Reports in IBM Spectrum Protect Plus provide users with status information. You can use *predefined reports*. Each predefined report has different options.

For more information about predefined reports, see *Types of reports* in the IBM Spectrum Protect Plus User's and Installation Guide.

In addition, you can create *custom reports* based on predefined reports.

Getting report information

You can get a list of reports and information for each. In the user interface, you can access reports on the following tabs: **Reports** and **Custom Reports**.

Method and URI

To get information about all reports, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/report
```

To get information about a specific report with `{reportIdStr}`, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/report/{reportIdStr}
```

Note: To get a {reportIdStr} by the report name, follow the instructions in "Getting a {reportIdStr}" on page 236.

Parameters

None.

Data

None.

Example: Get information about all reports

A Python snippet that is similar to the following example can be used to request information about all reports:

```
requests.get('https://' + spp_ipv4 + '/api/endeavour/report',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "total": 18,
    "page": 1,
    "reports": [
        {...}, {...}, ..., {...},
        {
            "links": {...},
```

```
            "name": "vSnap",
            "displayName": "vSnap Storage Utilization",
            "description": "Review the storage utilization for your vSnap Storag
    e Array.",
            "category": "Backup Storage Utilization",
            "categoryDisplayName": "Backup Storage Utilization",
            "custom": false,
            "creationTime": 1564685718121,
            "parentName": "vSnap",
            "parentDisplayName": "vSnap Storage Utilization",
            "rbacPath": "root:0/report:0/reportcategory:BackupStorageUtilization
    CategoryName/report:1013",
            "hasChildren": true,
            "id": "1013"
        },
        {...}, {...}, ..., {...},
        {
            "links": {...},
            "name": "vSnapStorageUtilization-Dallas1",
            "displayName": "vSnapStorageUtilization-Dallas1",
            "description": "A report of storage utilization for vSnap Dallas1",
            "category": "Backup Storage Utilization",
            "categoryDisplayName": "Backup Storage Utilization",
            "langType": "EN_US",
            "custom": true,
            "creationTime": 1575637231627,
            "parentId": "1013",
            "parentName": "vSnap",
            "parentDisplayName": "vSnap Storage Utilization",
            "rbacPath": "root:0/report:0/reportcategory:BackupStorageUtilization
    CategoryName/report:1102",
            "id": "1102"
        }
    ]
}
```

Deleting a custom report

You can delete a custom report.

Method and URI

To delete a custom report, use a DELETE method with a URI:

```
DELETE  https://{hostname|IPv4}/api/endeavour/report/{reportIdStr}
```

Note: To get the {reportIdStr} based on the report name, follow the information in “Getting a {reportIdStr}” on page 236.

Parameters

None.

Data

None.

Example: Delete a custom report

Assume that you added a custom report, vSnapStorageUtilization-Dallas1 (`{reportIdStr}` 1013), and you want to delete the custom report. Run the following command:

```
report_id_str = "1013"

requests.delete('https://' + spp_ipv4 + '/api/endeavour/report' + report_id_str,
    headers={...}, verify=...)
```

Ensure that you get a response with the HTTP status of 204 (No Content) and that you no longer see this custom report.

Scheduling a report

Schedule a report to run at a specific time and send the result to email recipients.

Method and URI

To schedule a report, use a POST method and a URI:

```
POST    https://{hostname|IPv4}/ngp/report
```

Parameters

Parameter 1: `action`

Start an action to schedule the report.

- **Value:** `schedule`
- **Type:** System string. Required. Available in the web user interface.

Data

Data 1: `userId`

Use this value when you want to run a report on behalf of someone else. For example, the user Sarah ({`userId`}: 1001} creates a session ID and sends a request to run a report without the `userId` key-value pair and her name appears in the report. Add *sppadmin* ({`userId`}: 1000) in the key-value pair and this username appears in the report.

Tip: To get a {userId} value based on its name, follow the information in "Getting a {userId}" on page 238.

- **Example value:** 1000
- **Type:** String.

Data 2: `resources`

Specify the reports to use. Use the following syntax:

```
[
   {
       "name":     {reportName 1},
       "id":       {reportIdStr 1},
       "rbacPath": {reportRbacPath 1},
       "href":     {reportHref 1}
    },
    ...,
    {...}
]
```

- **Example value:** A list that is similar to the following example can be used to specify the Database Backup History report:

```
[
   {
       "name":     "Database Backup History",
       "id":       "1012",
       "rbacPath": "root:0/report:0/reportcategory:ProtectionCategoryName/report
   :1012",
       "href":     "https://10.0.0.100/api/endeavour/report/1012"
    },
    ...,
    {...}
]
```

- **Type:** List (JSON object). Required. Available in the web user interface.

Data 2.1: `resources` > `name`

The name of the report.

- **Type:** String. Required. Available in the web user interface.

Data 2.2: `resources` > `id`

The {`reportIdStr`} of the report.

Tip: To get a {reportIdStr} based on the report name, follow the information in "Getting a {reportIdStr}" on page 236.

- **Type:** String. Required.

Data 2.3: `resources` > `rbacPath`

The {`reportRbacPath`} of the report.

Tip: To get a {reportRbacPath} based on the report name, follow the information in "Getting a {reportRbacpath}" on page 237.

- **Type:** String. Required.

Data 2.3: `resources` > `href`

The {`reportHref`} of the report.

- **Type:** String. Required.

Tip: To get a {reportHref} based on the report name, follow the information in "Getting a {reportHref}" on page 237.

- **Type:** JSON object. Required. Available in the web user interface.

Data 3: `trigger`

Define the frequency, start date and time, and the time zone for the web user interface. The structure of the value is the same as the value of `spec` > `subpolicy` > `trigger` for an SLA policy. For details, follow the instructions in "Adding an SLA policy" on page 48.

- **Type:** JSON object. Required. Available in the web user interface.

Data 4: `notification`

The email addresses of the recipients.

- **Example value:**

```
[
    "sarah.wiseman@bluemachines.com.invalid",
    "arthur.roberts@bluemachines.com.invalid"
]
```

- **Type:** List (string). Required. Available in the web user interface.

Data 5: `option`

Specify the options.

- **Example value:** Use the following value to get the scheduled report in PDF:

```
{
    "renderPdf":   true,
    "renderExcel": false,
    "renderWord":  false,
    "export":      false
}
```

- **Type:** JSON object. Available in the web user interface.

Data 5.1: `option` > `renderPDF`

Enable or disable to attach a copy of the report in PDF to the scheduled email.

- **Type:** Boolean. Available in the web user interface.

Data 5.2: `option` > `renderExcel`

Enable or disable to attach a copy of the report in Microsoft Excel format to the scheduled email.

- **Type:** Boolean. Available in the web user interface.

Data 5.3: `option` > `renderWord`

Enable or disable to attach a copy of the report in Microsoft Word format to the scheduled email.

- **Type:** Boolean. Available in the web user interface.

Example: Schedule a report

Sarah (`{userId: 1001}`) is based in Dallas where the time zone is US Central Time. First, she schedules one of the reports on a browser as below:

- **Report:** vSnap Storage Utilization (`{reportIdStr}`: 1013}
- **Frequency:** One time every day
- **Start Time:** 1 January 2020 at 0:00 AM US CST or 1577858400 in the UNIX timestamp format.
- **Recipients:**
  - sarah.wiseman@bluemachines.com.invalid
  - arthur.roberts@bluemachines.com.invalid
- **User:** sppadmin (`{userId}: 1000`)
- **Option:** Attach the PDF file to the scheduled report email.

A Python snippet that is similar to the following example can be used to scheule the report:

```
report_id = "1013"    # vSnap Storage Utilization

_params = {"action": "schedule"}

_data = f'''
    {{
        "userId":    "1000",
        "resources":    [{{
                            "name":     "vSnap Storage Utilization",
                            "id":       "{report_id}",
                        }}],
        "trigger":      {{
                            "frequency":    1,
                            "type":         "DAILY",
                            "activateDate": 1577858400000,
                            "timezone":     "America/Chicago",
                        }},
        "notification": [
                            "sarah.wiseman@bluemachines.com.invalid",
                            "arthur.roberts@bluemachines.com.invalid"
                        ],
        "option" {{
            "renderPdf":   true,
            "renderExcel": false,
            "renderWord":  false,
            "export":      false
        }}

     }}'''

requests.post('https://' + spp_ipv4 + '/ngp/report',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created).

```
{
    "statusCode": 201,
    "response": [
```

```
        {
            "links": {...},
            "name": "Report_vSnap Storage Utilization",
            "type": "Schedule",
            "category": "SCHEDULE",
            "properties": {
                "activateDate": 1577858400000,
                "deactivateDate": null,
                "frequency": 1,
                "type": "DAILY",
                "dowList": [
                    false,
                    ...
                ],
                "domList": [
                    false,
                    ...
                ]
            },
            "rbacPath": "trigger:0/trigger:1020",
            "onHold": false,
            "id": "1020"
        },
        {
            "links": {...},
            "name": "Report_vSnap Storage Utilization",
            "type": "analyze",
            "subType": "report",
            "serviceId": "serviceprovider.report",
            "description": "",
            "version": "3.1",
            "rbacPath": "root:0/policy:0/policytype:serviceprovider.report/polic
    y:1023",
            "tenantId": 1000,
            "creationTime": 1576173685532,
            "lastUpdated": 1576177184653,
            "spec": {
                "source": [
                    {
                        "href": "https://10.0.0.100/api/endeavour/report/1013",
                        "name": "vSnap Storage Utilization",
                        "id": "1013",
                        "parentId": "",
                        "tags": {}
                    }
                ],
                "option": {
                    "renderPdf": true,
                    "renderWord": false,
                    "renderExcel": false,
                    "export": false
                },
                "notification": [
                    {
                        "href": "https://10.0.0.100/api/smtp/1001",
                        "type": "email",
                        "recipients": [
                            "sarah.wiseman@bluemachines.com.invalid",
                            "arthur.roberts@bluemachines.com.invalid"
                        ]
                    }
                ]
            },
            "logicalDelete": false,
            "id": "1023"
        }
    ]
}
```

Creating a custom report

You can modify predefined reports with custom parameters in IBM Spectrum Protect Plus and save the customized reports.

Method and URI

To create a custom report, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/api/endeavour/report
```

Parameters

None.

Data

Data 1: `name`

The name of the custom report.

- **Example value:** vSnapStorageUtilization-Dallas1
- **Restriction:** Report names can include alphanumeric characters and the following symbols:

```
$-_.+!*'()
```

Spaces are not permitted.

- **Type:** String. Required. Available in the web user interface.

Data 2: `description`

The description of the custom report.

- **Example value:** A report about storage utilization for vSnap Dallas1
- **Type:** String. Available in the web user interface.

Data 3: `paramValues`

The options of the custom report. The available options depend on the original predefined report. For details, follow the instructions in “Report parameters” on page 166.

- **Example value:** The following value defines example options that are available in the predefined report, vSnap Storage Utilization, and custom reports based on it:

```
{
    "vSnap":              ["(All)"],
    "IncludeReplicaVols": false,
    "VMsPerStorage":      false
}
```

- **Type:** JSON object. Required. Available in the web user interface.

Data 4: `parentIid`

The `{reportIdStr}` of the original predefined report.

- **Example value:** 1013
- **Type:** String. Required. Available in the web user interface.

Tip: To get a {reportIdStr} by the report name, follow the information in “Getting a {reportIdStr}” on page 236.

Example: Create a custom report

Assume that you want to create the following custom report:

- **Name:** vSnapStorageUtilization-Dallas1
- **Description:** A report about storage utilization for vSnap Dallas1
- **Options:**
  - **Exclude Replica Destination Volume:** true
  - **Show Resources protected per vSnap Storage:** true
  - **vSnap server:** 10.0.1.1 (`{vSnapId}` 2000)
- **Original predefined report:** vSnap Storage Utilization (`{reportIdStr}` 1013)

A Python snippet that is similar to the following example can be used to create a custom report based on the vSnap Storage Utilization report:

```
_data = f'''{{
    "name":        "vSnapStorageUtilization-Dallas1",
    "description": "A report of storage utilization for vSnap Dallas1",
    "paramValues": {{
        "vSnap":              ["2000"],
        "IncludeReplicaVols": true,
        "VMsPerStorage":      true
    }},
    "parentId":"1013"
}}'''

requests.post(sppUrl + '/api/endeavour/report',
     headers=...,  data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created). Review the response to ensure that the custom report was added.

```
{
    "links": {...},
    "name": "vSnapStorageUtilization-Dallas1",
    "displayName": "vSnapStorageUtilization-Dallas1",
    "description": "A report of storage utilization for vSnap Dallas1",
    "categoryDisplayName": "Backup Storage Utilization",
    "langType": "EN_US",
    "custom": true,
    "creationTime": 1577082702705,
    "paramValues": {
        "vSnap": [
            "2000"
        ],
        "IncludeReplicaVols": true,
        "VMsPerStorage": true
    },
    "parentId": "1013",
    "parentName": "vSnap",
    "rbacPath": "root:0/report:0/reportcategory:BackupStorageUtilizationCategory
   Name/report:1101",
    "id": "1101"
}
```

Report parameters

"Creating a custom report" on page 165

As mentioned in, you must specify options when you create a custom report that is based on a predefined report. Every predefined report has different options. Specify new options by using the `paramValues` key. For more information about predefined reports, see the IBM Spectrum Protect Plus User's and Installation Guide.

vSnap Storage Utilization report

Review the storage utilization of your vSnap servers, including the availability status, free space, and used space. The vSnap Storage Utilization report displays both an overview of your vSnap servers and a detailed view of the individual virtual machines and databases that are protected on each vSnap server.

Category

Backup Storage Utilization

paramValues

Assume that you want to use the following options for the Protected and Unprotected VMs report:

- **vSnap storage:** All
- **Exclude replica Destination Volumes:** No
- **Show resources protected per vSnap storage:** No

To use the listed options, the `paramValues` key will have the following JSON object:

```
{
    "vSnap":              ["2000"],
    "IncludeReplicaVols": false,
    "VMsPerStorage":      false
}
```

paramValues 1: `vSnap`

To select all vSnap servers, use `["(All)"]`. To select specific vSnap servers, enter their `{storageId}`.

- **Example value:** `["2000", "2001"]`
- **Type:** Array. Required. Avaiable in the web user interface.

Tip: To get a {storageId} value based on its name, follow the information in "Getting a {storageId}" on page 242.

paramValues 2: `IncludeReplicaVols`

Exclude replica destination volumes.

- **Type:** Boolean. Required. Avaiable in the web user interface.

paramValues 3: `VMsPerStorage`

Show resources protected per vSnap server.

- **Type:** Boolean. Required. Avaiable in the web user interface.

Protected and Unprotected VMs report

The Protected and Unprotected VMs report shows the protection status of your virtual machines. The report displays the total number of virtual machines added to the IBM Spectrum Protect Plus inventory before backup jobs are started.

Category

Protection

paramValues

Assume that you want to use the following options for the Protected and Unprotected VMs report:

- **Virtualized system type:** VMware

- **Hypervisors or Amazon EC2 accounts:** BlueMachines vCenter - Dallas (`{hypervisorHypervisorId} 1001`)
- **Hide unprotected virtual machines:** No
- **Show only the virtual machines with object storage copy backups:** No
- **Tags:** Sales

To use the listed options, the `paramValues` key will have the following JSON object:

```
{
    "NodeCascadingParameterGroup-HypervisorType": "vmware",
    "NodeCascadingParameterGroup-Hypervisor":     ["1001"],
    "UnprotectedVMs":                             false,
    "OffloadedVMs":                               false,
    "tags":                                       ["Dallas-Production"]
}
```

paramValues 1: `NodeCascadingParameterGroup-HypervisorType`

Specify a type of the virtualized systems:

- **Value:** Use one of the following values:

Virtualized system type	NodeCascadingParameterGroup-HypervisorType
All	(All)
VMware	vmware
Hyper-V	hyperv
Amazon EC2	awsec2

- **Type:** System string. Required. Available in the web user interface.

paramValues 2: `NodeCascadingParameterGroup-Hypervisor`

Specify the hypervisor or the Amazon EC2 account.

Tip: To get a {hypervisorHypervisorId} by hostname or IP, follow the information in "Getting a {hypervisorHypervisorId} " on page 227.

- **Type:** List (string). Required. Available in the web user interface.

paramValues 3: `UnprotectedVMs`

Indication of whether the result omits unprotected virtual machines.

- **Type:** Boolean. Required. Available in the web user interface.

paramValues 4: `OffloadedVMs`

Indication of whether the result shows only the virtual machines with object storage copy backups.

- **Type:** Boolean. Required. Available in the web user interface.

paramValues 5: `tags`

Tag names for VMware instances. Or, you can use `(All)` to select all tags.

- **Example:** `["Dallas-Production"]`
- **Type:** List (string). Required. Available in the web user interface.

Configuration report

You can use the predefined report, Configuration, or any custom report based on it.

paramValues

paramValues 1 `ConfigurationType`

A list that includes the following system string values:

ConfigurationType	Description
(All)	All options
Application Servers	Application Servers
Backup Storage - Cloud	Backup Storage - Cloud
Backup Storage - Disk	Backup Storage - Disk
Backup Storage - Repository Server	Backup Storage - Repository Server
Hypervisors	Hypervisors
LDAP	LDAP
SMTP	SMTP
VADP Proxy	VADP Proxy

Example value: Use the following list to enable Application Servers, Backup Storage - Disk, and Hypervisors.

```
[
    "Application Servers",
    "Backups Storage - Disk",
    "Hypervisors"
]
```

Type: List (system string). Required. Available in the web user interface.

Additional topics for vSnap Storage Utilization

You can complete additional operations for the report vSnap Storage Utilization.

Protected virtual machines

You can get basic information about protected virtual machines.

Method and URI

To get information about all protected virtual machines, use a GET method and a URI:

```
GET     https://{hostname|IPv4}/api/endeavour/catalog/recovery/hypervisorvm
```

Parameters

Parameter 1: `embed`

Specify the type of action.

- **Value:** `(children(properties))`
- **Type:** System string. Required.

Data

None.

Example: Get information about protected virtual machines

Assume that you want to get information about protected virtual machines.

A Python snippet that is similar to the following example can be used to get a list of protected virtual machines and detailed information about each:

```
_params = {"embed": "(children(properties))"}

requests.post('https://' + spp_ipv4
    + '/api/endeavour/catalog/recovery/hyipervisorvm',
    headers={...}, params=_params, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "queryElapsedTime": 11,
    "id": "hypervisorvm",
    "name": "hypervisorvm",
    "type": "folder",
    "catalogTime": null,
    "catalogProvider": "recovery",
    "comment": null,
    "tags": null,
    "documentType": null,
    "page": 1,
    "properties": {},
    "children": [
        {
            "links": {...},
            "id": "5d551d2816186f1eeea46718",
            "name": "sales-sql",
            "type": "document",
            "catalogTime": 1565871979100,
            "catalogProvider": "recovery",
            "comment": null,
            "tags": null,
            "documentType": "vm",
            "metadataPath": "/MS:1002/VM:8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9",
            "properties": {
                "name": "sales-sql",
                "comment": null,
                "creatorId": 0,
                "creationTime": null,
                "type": "vm",
                "flags": 0,
                "pk": "8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9",
                "sessionId": 1565871979100,
                "objType": null,
                "jobId": "1014",
                "host": null,
                "protectionInfo": {
                    "policyName": "hyperv_Gold",
                    "subPolicyName": "VADP Backup(2000)",
                    "subPolicyId": "2000",
                    "storageProfileId": 2000,
                    "storageProfileName": "Gold",
                    "subPolicyLabel": null,
                    "subPolicyType": "BACKUP",
                    "subPolicySubType": "vsnap",
                    "originalSubPolicyType": null,
                    "protectionTime": 1565873191165,
                    "lastErrorMessage": null,
                    "recoveryPoint": "1365d24f-f8ca-4e29-9064-f487e8906701",
                    "mirrorLabel": null,
                    "snapLabels": null,
                    "snapShotName": null,
                    "protectedVolumePks": [],
                    "flashCopyIds": null,
                    "emcVnxSnapshotIds": null,
```

```
                    "cloudSnapshotIds": null,
                    "storageSnapshots": [
                        {
                            "storageType": "vsnap",
                            "storageSnapshotId": "2000.snapshot.87",
                            "snapshotReplicationId": null,
                            "snapshotVersionId": "17374895456114910019",
                            "storageId": "2000",
                            "storageVolumeId": "2000.volume.20",
                            "contentType": null
                        },
                        {
                            "storageType": "vsnap",
                            "storageSnapshotId": "2000.snapshot.87",
                            "snapshotReplicationId": null,
                            "snapshotVersionId": "17374895456114910019",
                            "storageId": "2000",
                            "storageVolumeId": "2000.volume.20",
                            "contentType": null
                        }
                    ],
                    "maskSnapshotIds": null,
                    "storageSiteIds": [
                        "1000"
                    ],
                    "primary": null,
                    "baseBackup": false,
                    "retentionInfo": {
                        "type": "days",
                        "count": 7,
                        "expiration": 1566478113751
                    },
                    "skippedVolumePks": [],
                    "protectionMethod": "VMWARE_CBT",
                    "resourceType": null,
                    "destinationPk": null,
                    "destinationInfo": {
                        "destinationFileName": null,
                        "vmx": null,
                        "vmxf": null,
                        "vmxBackup": null,
                        "vmxfBackup": null,
                        "configFile": "Virtual Machines\\8989CFF1-6F16-4E2B-9FCC
    -9F8C8CD719C9.VMCX",
                        "extendedConfigFile": null,
                        "configFileBackup": null,
                        "extendedConfigFileBackup": null,
                        "backupPath": "sales-sql_8ee9123",
                        "destinationVolumeKey": "ae85b9df-5739-4de9-a287-b12ae2a
    030b9"
                    },
                    "storageType": [
                        "VSNAP"
                    ],
                    "backupMetadata": null,
                    "backupDocumentKey": null,
                    "rmanTag": null,
                    "recoveryTransactionId": null,
                    "masked": false,
                    "forceBase": false,
                    "dbServerTime": null,
                    "incrementalSnapshot": false,
                    "isCGCreatedByEcx": false,
                    "transferDuration": 66668,
                    "transferSize": 9256960,
                    "storageInfo": "{\"localhost\":[\"spp_1014_2000_16c946367fd_
    _16c94639564_15_vsnap\"]}",
                    "sppoFromVSnap": false,
                    "excludedFromBackup": false,
                    "dummyObject": false,
                    "backupPathsInfos": [
                        {
                            "destStorageVolumePk": "2000.volume.20",
                            "backupPaths": [
                                "sales-sql_8ee9123"
                            ]
                        },
                        {
                            "destStorageVolumePk": "2000.volume.20",
                            "backupPaths": [
                                "sales-sql_8ee9123"
                            ]
```

```
                    }
                ],
                "baseBackupSize": 9295626240,
                "combinedDelta": 0
            },
            "parentPk": "2d60ba76-9161-3492-ae5d-80f0b067760a",
            "providerNodeId": "1002",
            "providerUniqueId": null,
            "siteId": null,
            "autoProvisionedsubPolicyName": null,
            "rbacPath": "rbac:0/hypervisor.all:0/hypervisor.type:hyperv/hype
rvisor:1002/vm:8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9",
            "rbacPathId": 186,
            "location": "/dallas-hv1",
            "metadataPath": "/MS:1002/VM:8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C
9",
            "isProtected": false,
            "inHLO": false,
            "hypervisorType": "hyperv",
            "hypervisorKey": "1002",
            "hypervisorManagementServerID": "1002",
            "nativeKey": null,
            "cloudType": null,
            "tags": [],
            "tagCategories": [],
            "metadataPathVmHost": "/MS:1002/VM:8989CFF1-6F16-4E2B-9FCC-9F8C8
CD719C9/CLUST:2d60ba76-9161-3492-ae5d-80f0b067760a/HOST:8a88cacf-80f5-308d-bdaf
-f2939920ee87/VM:8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9",
            "rbacPathVmHost": "rbac:0/hypervisor.all:0/hypervisor.type:hyper
v/hypervisor:1002/vm:8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9/vcluster:2d60ba76-916
1-3492-ae5d-80f0b067760a/vhost:8a88cacf-80f5-308d-bdaf-f2939920ee87/vm:8989CFF1
-6F16-4E2B-9FCC-9F8C8CD719C9",
            "rbacPathIdVmHost": 187,
            "alternatePaths": [],
            "isTemplate": false,
            "isEncrypted": false,
            "powerSummary": {
                "powerState": "ON",
                "lastPowerOnTime": 1564750421466,
                "lastPowerOffTime": 0,
                "lastSuspendTime": 0,
                "uptime": 0
            },
            "storageSummary": {
                "commited": 9295626240,
                "uncommited": 45465206784,
                "shared": 0
            },
            "hypervisorHostKey": "8a88cacf-80f5-308d-bdaf-f2939920ee87",
            "hypervisorHostName": "dallas-hv1",
            "volumes": [
                {
                    "name": "csvvol1",
                    "key": "2b7532f9-7f2e-4457-ab79-160b631fc66f",
                    "isConfigVolume": false
                },
                {
                    "name": "csvvol2",
                    "key": "9dab0f74-ace3-4d1a-bb5a-07e7ed10c185",
                    "isConfigVolume": true
                }
            ],
            "uniqueId": "8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9",
            "configFilePath": "Virtual Machines\\8989CFF1-6F16-4E2B-9FCC-9F8
C8CD719C9.VMCX",
            "fileDirPath": "SSwin2k12",
            "cluster": {
                "name": "dallas-hv1",
                "key": "2d60ba76-9161-3492-ae5d-80f0b067760a"
            },
            "cpu": 1,
            "coresPerCpu": 1,
            "memory": 1073741824,
            "vmVersion": "8.0",
            "vmStorageUtilization": 0,
            "vmDestVolDedupCompressionInfo": [],
            "configInfo": {
                "name": null,
                "comment": null,
                "creatorId": 0,
                "creationTime": null,
                "type": null,
```

```
                        "flags": 0,
                        "pk": null,
                        "sessionId": null,
                        "objType": null,
                        "jobId": null,
                        "host": null,
                        "providerNodeId": null,
                        "providerUniqueId": null,
                        "siteId": null,
                        "autoProvisionedsubPolicyName": null,
                        "isProtected": false,
                        "inHLO": true,
                        "hypervisorType": null,
                        "hypervisorKey": null,
                        "hypervisorManagementServerID": null,
                        "nativeKey": null,
                        "cloudType": null,
                        "tags": null,
                        "tagCategories": null,
                        "vmId": null,
                        "osName": null,
                        "osVersion": null,
                        "hostName": null,
                        "vmToolsStatus": null,
                        "guestNetworkInfo": []
                    }
                }
            },
            {...}, ..., {...}
        ]
    }
```

Base backup sizes

You can get a base backup size in bytes for each protected virtual machine by using the primary key (pk).

Tip: To get the primary key (pk) as well as other detailed information about each protected virtual machine, follow the instructions in this section. To get the sum of data sizes of all backup snapshots for each protected virtual machines, follow the information in “Incremental backup sizes” on page 175.

Method and URI

To get a base backup size for each protected virtual machine, use a POST method and a URI:

```
POST    https://{hostname|IPv4}/api/endeavour/catalog/recovery/hypervisorvm
```

Parameters

Parameter 1: `action`

Invoke aggregate functions. For more information about aggregate functions, follow the instructions in “Aggregate functions” on page 246.

- **Value:** `aggregate`
- **Type:** Array. Required. Available in the web user interface.

Parameter 2: `filter`

- **Value:**

```
[
    {
        "property": "sessionId",
        "op":       ">",
        "value":    "0"
    },
    {
        "property": "protectionInfo.baseBackup",
        "op":       "=",
        "value":    "true"
```

```
    }
]
```

- **Type:** Array. Required. Available in the web user interface.

Data

Data 1: op

Use the `MAX()` function for the `protectionInfo.transferSize` values. For more information about the `MAX()` function, follow the instructions in “MAX()” on page 247.

- **Value:**

```
[
    {
       "operation":  "max",
       "fieldname":  "protectionInfo.transferSize",
       "outputname": "baseBackupSize"
    }
]
```

- **Type:** Array. Required. Available in the web user interface.

Data 2: `group`

Use the GROUP clause with the primary key values. For more information about the GROUP clause, follow the instructions in “GROUP clause” on page 248.

- **Value:** `["pk"]`
- **Type:** Array. Required. Available in the web user interface.

Data 3: `copytype`

- **Value:** `versiononly`
- **Type:** System string. Required.

Example: Get information about the base backup size for virtualized system protection

Assume that you have two protected virtual machines:

- sales-americas
  - **Primary key:** 92647504afd9005226aef77bc0b98abb
  - **Base backup size:** 100 GB (107,374,182,400 bytes)
  - **Incremental backup snapshot sizes:** 10 GB (10,737,418,240 bytes, 1 GB x 10 backup snapshots)
  - **Total backup size:** 110 GB
- sales-sql
  - **Primary key:** 8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9
  - **Base backup size:** 1 TB (1,073,741,824,000 bytes)
  - **The sum of all incremental backup snapshots:** 1 TB x 1 backup snapshot
  - **Total backup size:** 2 TB

A Python snippet that is similar to the following example can be used to get a list of protected virtual machines and a base backup size for each. In this example, the value will be shown with the output key name `baseBackupSize`.

```
_params = {
    "action": "aggregate",
    "filter": str([
        {
```

```
            "property": "sessionId",
            "op":       ">",
            "value":    "0"
        },
        {
            "property": "protectionInfo.baseBackup",
            "op":       "=",
            "value":    "true"
        }
    ])
}

_data = f'''{{
    "op":    [
        {{
            "operation":  "max",
            "fieldname":  "protectionInfo.transferSize",
            "outputname": "baseBackupSize"
        }}
    ],
    "group": [
        "pk"
    ],
    "copytype": "versiononly"
}}'''


requests.post('https://' + spp_ipv4
    + '/api/endeavour/catalog/recovery/hypervisorvm',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "results": [
        {
            "_id": {
                "pk": "2cc67ed7057bf949ce068f477a1ee6f5"
            },
            "baseBackupSize": 107374182400
        },
        {
            "_id": {
                "pk": "8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9"
            },
            "baseBackupSize": 1073741824000
        },
        {...}, ..., {...}
    ]
}
```

Incremental backup sizes

You can get an incremental backup size for each protected virtual machine by using its primary key (pk).

Tip: To get the primary key (pk) as well as other detailed information about each protected virtual machine, follow the instructions in “Protected virtual machines” on page 169. To get the data size of the base backup for each protected virtual machine, follow the instructions in “Base backup sizes” on page 173.

Method and URI

To get an incremental backup size for each protected virtual machine, use a POST method and a URI:

```
POST    https://{hostname|IPv4}/api/endeavour/catalog/recovery/hypervisorvm
```

Parameters

Parameter 1: `action`

Invoke aggregate functions. For more information about aggregate functions, follow the instructions in “Aggregate functions” on page 246.

- **Value:** `aggregate`
- **Type:** System string. Required. Available in the web user interface.

Parameter 2: `filter`

Optionally, you may use a filter to specify the type of virtualized systems. If this parameter is not used, the request counts all protected virtual machines regardless of the type of virtualized systems. You can use the `filter` operation parameters that are described in “Filter” on page 243.

- **Value:**

```
[
    {
        "property": "sessionId",
        "op":       ">",
        "value":    "0"
    },
    {
        "property": "protectionInfo.baseBackup",
        "op":       "=",
        "value":    "false"
    }
]
```

- **Type:** Array. Required. Available in the web user interface.

Data

Data 1: op

Use the SUM() function for the `protectionInfo.transferSize` values. For more information about the SUM() function, follow the instructions in

“SUM()” on page 247

.

- **Value:**

```
[
    {
       "operation":  "sum",
       "fieldname":  "protectionInfo.transferSize",
       "outputname": "incrBackupSize"
    }
]
```

- **Type:** Array. Required. Available in the web user interface.

Data 2: `group`

Use the GROUP clause with the primary key values. For more information about the GROUP clause, follow the instructions in

“GROUP clause” on page 248

.

- **Value:** `["pk"]`
- **Type:** Array. Required. Available in the web user interface.

Data 3: `copytype`

- **Value:** `versiononly`
- **Type:** System string. Required.

Example: Get information about incremental backup sizes

Assume that you have two protected virtual machines:

- sales-americas
  - **Primary key:** 92647504afd9005226aef77bc0b98abb
  - **Base backup size:** 100 GB (107,374,182,400 bytes)
  - **Incremental backup snapshot sizes:** 10 GB (10,737,418,240 bytes, 1 GB x 10 backup snapshots)
  - **Total backup size:** 110 GB (115,343,360 bytes)
- sales-sql
  - **Primary key:** 8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9
  - **Base backup size:** 1 TB (1,073,741,824,000 bytes)
  - **The sum of all incremental backup snapshots:** 1 TB x 1 backup snapshot
  - **Total backup size:** 2 TB

The Python code snippet below will return a list of protected virtual machines and their primary key and incremental backup size. In this example, the value will be shown with the key name `baseBackupSize`.

```
_params = {
    "action": "aggregate",
    "filter": str([
        {
            "property": "sessionId",
            "op":       ">",
            "value":    "0"
        },
        {
            "property": "protectionInfo.baseBackup",
            "op":       "=",
            "value":    "false"
        }
    ])
}

_data = f'''{{
    "op":    [
        {{
            "operation":  "sum",
            "fieldname":  "protectionInfo.transferSize",
            "outputname": "incrBackupSize"
        }}
    ],
    "group": [
        "pk"
    ],
    "copytype": "versiononly"
}}'''

requests.post('https://' + spp_ipv4
    + '/api/endeavour/catalog/recovery/hypervisorvm',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "results": [
        {
            "_id": {
                "pk": "8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9"
            },
            "incrBackupSize": 1073741824000
        },
```

```
        {
            "_id": {
                "pk": "92647504afd9005226aef77bc0b98abb"
            },
            "incrBackupSize": 10737418240
        },
        {...}, ..., {...}
    ]
}
```

Recovery points

You can get the number of restore points that are available (a recovery point) for each protected virtual machine with its primary key (pk).

Tip: To get the primary key (pk) as well as other detailed information about each protected virtual machine, follow the information in “Protected virtual machines” on page 169.

Method and URI

To get a recovery point for each protected virtual machine, use a POST method and a URI:

```
POST    https://{hostname|IPv4}/api/endeavour/catalog/recovery/hypervisorvm
```

Parameters

Parameter 1: `action`

Invoke aggregate functions. For more information about aggregate functions, follow the instructions in “Aggregate functions” on page 246.

- **Value:** `aggregate`
- **Type:** Array. Required. Available in the web user interface.

Parameter 2: `filter`

- **Value:**

```
[
    {
        "property": "sessionId",
        "op":       ">",
        "value":    "0"
    }
]
```

- **Type:** Array. Required. Available in the web user interface.

Data

Data 1: op

Use the `COUNT()` function for the pk values. For more information about the `COUNT()` function, follow the instructions in “COUNT()” on page 247.

- **Value:**

```
[
    {
       "operation":  "count",
       "fieldname":  "pk",
       "outputname": "count"
    }
]
```

- **Type:** Array. Required. Available in the web user interface.

Data 2: `group`

Use the GROUP clause with the primary key values. For more information about the GROUP clause, follow the instructions in "GROUP clause" on page 248.

- **Value:** `["pk"]`
- **Type:** Array. Required. Available in the web user interface.

Data 3: `copytype`

- **Value:** `versiononly`
- **Type:** System string. Required.

Example: Get information about recovery points

Assume that you have two protected virtual machines:

- **sales-americas:**
  - **Primary key:** 92647504afd9005226aef77bc0b98abb
  - **Base backup size:** 100 GB (107,374,182,400 bytes)
  - **Incremental backup snapshot sizes:** 10 GB (10,737,418,240 bytes, 1 GB x 10 backup snapshots)
  - **Total backup size:** 110 GB
- **sales-sql:**
  - **Primary key:** 8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9
  - **Base backup size:** 1 TB (1,073,741,824,000 bytes)
  - **The sum of all incremental backup snapshots:** 1 TB x 1 backup snapshot
  - **Total backup size:** 2 TB

A Python snippet that is similar to the following example can be used to get a list of protected virtual machines and their primary keys and recovery points. In this example, the value will be shown with the key name `count`.

```
_params = {
    "action": "aggregate",
    "filter": str([
        {
            "property": "sessionId",
            "op":       ">",
            "value":    "0"
        }
    ])
}

data = f'''{{
    "op":       [
                    {{
                        "operation":  "count",
                        "fieldname":  "pk",
                        "outputname": "count"
                    }}
                ],
    "group":    [
                    "pk"
                ],
    "copytype": "versiononly"
}}'''

requests.post('https://' + spp_ipv4
    + '/api/endeavour/catalog/recovery/hypervisorvm',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK). As you can see, the protected virtual machine sales-americas has ten backup snapshots (recovery points) and sales-sql has one snapshot.

```
{
    "links": {...},
    "results": [
        {
            "_id": {
                "pk": "8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9"
            },
            "count": 1
        },
        {
            "_id": {
                "pk": "92647504afd9005226aef77bc0b98abb"
            },
            "count": 10
        },
        {...}, ..., {...}
    ]
}
```

Analytics

You can analyze the system operations of IBM Spectrum Protect Plus by viewing the dashboard in the user interface and by viewing reports. However, some types of analytical information are available only by using the REST API.

Getting backup history for a virtual machine

You can get virtual machine backup history. This action is available only in the REST API.

URI and Method

To get information about virtual machine backup history, use a GET method and a URI:

```
GET     https://{hostname|IP}/ngp/analytics/vmbackuphistory
```

Parameters

None.

Data

None.

Example: Get virtual machine backup history

A Python snippet that is similar to the following example can be used to request information about virtual machine backup history.

```
requests.get('https://' + spp_ipv4 + '/ngp/analytics/vmbackuphistory',
    headers={...}, verify=spp_verify)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "instances": [
        {
            "pk": "8989CFF1-6F16-4E2B-9FCC-9F8C8CD719C9",
            "VM": "Sales-Win2016-001",
```

```
                "tags": [],
                "SLAPolicy": "Gold",
                "ProtectionTime": 1617235200111,
                "hypervisor": "dallas-hv1.bluemachines.com.invalid",
                "hypervisorType": "Hyper-V",
                "Status": "Success",
                "BackupStorage": "localhost",
                "BackupDetails": {
                    "BackupData": 416746424,
                    "BackupDuration": 237536
                },
                "TotalRuns": 2
            },
            {...}, ..., {...}
        ]
    }
```

In the previous response, the `ProtectionTime` value represents the date and time when this backup job was completed. This timestamp uses the UNIX format followed by three digits that represent milliseconds. For example, the value 1617235200111 represents 1 April 2021 at 0:00.111 AM UTC.

The `BackupDetails` > `BackupData` value represents the total size of the backup snapshot in bytes. For example, the value 416746424 represents approximately 397 MB.

The `BackupDuration` value represents the time in milliseconds to complete the backup job. For example, the value 237536 represents 3 minutes and 57.536 seconds.

Getting database backup history

You can get database backup history. This action is available only in the REST API.

Method and URI

To get database backup history, use a GET method and a URI:

```
GET     https://{hostname|IP}/ngp/analytics/dbbackuphistory
```

Parameters

None.

Data

None.

Example: Get database backup history

A Python snippet that is similar to the following example can be used to request information about database backup history.

```
requests.get('https://' + spp_ipv4 + '/ngp/analytics/dbbackuphistory',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "instances": [
        {
            "pk": "7540bc5013b15e4940ebbb908c4aaf87",
            "parentPk": "34ea1797e3fd1d00d33b349517057d1d",
            "DB": "SALES-AMERICAS1",
            "SLAPolicy": "Gold",
            "ProtectionTime": 1570380381334,
            "Status": "Failed",
            "BackupStorage": "",
```

```
            "BackupDetails": {
                "FailureReason": "[Backup failed: Backup failed, Backup of all d
    atabases failed :All databases failed]"
            },
            "TotalRuns": 392
        },
        {...}, ..., {...}
    ]
}
```

Getting virtual machine backup compliance

You can get information about virtual machine backup compliance. This action is available only in the REST API.

URI and Method

To get information about virtual machine backup compliance, use a GET method and a URI:

```
GET     https://{hostname|IP}/ngp/analytics/vmbackupslacom
```

Parameters

None.

Data

None.

Example: Get information about virtual machine backup compliance

A Python snippet that is similar to the following example can be used to request information about virtual machine backup compliance.

```
requests.get('https://' + spp_ipv4 + '/ngp/analytics/vmbackupslacom',
    headers={...}, verify=spp_verify)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "backup2vsnap": {
        "compliance": [],
        "ucompliance": [
            {
                "VM": "Sales-Windows2016-005",
                "SLAPolicy": "Gold",
                "Hypervisor": "win2016enghv.bluemachines.com.invalid",
                "HypervisorType": "hyperv",
                "ProtectionTime": 1565873191165,
                "Location": "/win2016enghv",
                "Frequency": 4,
                "Schedule": "HOURLY",
                "FailureReason": "Defined RPO for SLA Policy Gold to run Every 4
    hour(s) not meet."
            },
            {
                "VM": "Sales-RedHatLinux-004",
                "SLAPolicy": "Gold",
                "Hypervisor": "10.0.0.10",
                "HypervisorType": "vmware",
                "ProtectionTime": 1573345610009,
                "Location": "/vmware-dallas1.bluemachines.com.invalid/Dallas -
   DataCenter1/Lab folder",
                "Frequency": 4,
                "Schedule": "HOURLY",
                "FailureReason": "Defined RPO for SLA Policy Gold to run Every 4
    hour(s) not meet."
            }
```

```
            ]
        },
        "replication": {
            "compliance": [],
            "ucompliance": []
        },
        "offload": {
            "compliance": [],
            "ucompliance": []
        }
    }
```

Getting database backup compliance

You can get information about database backup compliance. This action is available only in the REST API.

Method and URI

To get information about database backup compliance, use a GET method and a URI:

```
GET     https://{hostname|IP}/ngp/analytics/dbbackupslacom
```

Parameters

None.

Data

None.

Example: Get database backup compliance

A Python snippet that is similar to the following example can be used to request information about database backup compliance.

```
requests.get('https://' + spp_ipv4 + '/ngp/analytics/dbbackupslacom',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "backup2vsnap": {
        "compliance": [],
        "ucompliance": [
            {
                "DB": "Bluemachines-Oracle-SalesDB",
                "SLAPolicy": "Gold",
                "ProtectionTime": 1567168142000,
                "Frequency": 4,
                "Schedule": "HOURLY",
                "FailureReason": "[Backup failed: Backup failed, Backup of all d
   atabases failed :All databases failed]"
            },
            ..., {...}
        ]
    },
    "replication": {
        "compliance": [],
        "ucompliance": []
    },
    "offload": {
        "compliance": [],
        "ucompliance": []
    }
}
```

Counting virtual machines

Method and URI

To count all protected and unprotected virtual machines registered by IBM Spectrum Protect Plus, use a POST method and a URI:

```
POST    https://{hostname|IP}/api/endeavour/catalog/hypervisor/vm
```

Parameters

Parameter 1: `action`

Invoke aggregate functions. For more information about aggregate functions, follow the instructions in "Aggregate functions" on page 246.

- **Value:** `aggregate`
- **Type:** Array. Required. Available in the web user interface.

Parameter 2: `filter`

Optionally, you may use a filter to specify the virtualized system type: VMware vCenter, Hyper-V, or Amazon EC2. If this parameter is not used, IBM Spectrum Protect Plus counts all virtual machines. You can use the `filter` operation parameters that are described in "Filter" on page 243.

- **Example value:** To search within instances that are associated with no SLA policies, use the following filter parameter:

```
[
    {
        "op":       "=",
        "property": "hypervisorType",
        "value":    "vmware"
    }
]
```

Choose one of the following virtualized systems and use the value in the filter.

Virtualized system	Value
VMware vCenter	vmware
Hyper-V	hyperv
Amazon EC2	awsec2

- **Type:** JSON object. Available in the web user interface.

Data

Data 1: op

Use the `COUNT()` function to count the pk objects. For more information about the `COUNT()` function, follow the instructions in "COUNT()" on page 247. Name the output variable `count`.

- **Value:**

```
[
    {
        "operation":  "count",
        "fieldname":  "pk",
        "outputname": "count"
    }
]
```

- **Type:** Array. Required. Available in the web user interface.

Example: Count virtual machines that are managed on VMware vCenter

A Python snippet that is similar to the following example can be used to count the virtual machines that are managed by VMware vCenter Server, and registered with IBM Spectrum Protect Plus. The number includes both protected virtual machines and unprotected virtual machines.

```
hypevisor_type = "vmware"     # VMware vCenter
# hypevisor_type = "hyperv"   # Hyper-V
# hypevisor_type = "awsec2"   # Amazon EC2

_params = {
    "action": "aggregate",
    "filter": f'''[
        {{
             "op":       "=",
             "property": "hypervisorType",
             "value":    {hypervisor_type}
        }}
    ]'''
}

_data = f'''{{
    "op": [
        {{
            "operation":  "count",
            "fieldname":  "pk",
            "outputname": "count"
        }}
    ],
}}'''

requests.post('https://' + spp_ipv4
    + '/api/endeavour/catalog/hypervisor/vm',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "results": [
        {
            "_id": null,
            "count": 600
        }
    ]
}
```

Counting databases

Method and URI

To count all protected and unprotected databases registered by IBM Spectrum Protect Plus, use a POST method and a URI:

```
POST    https://{hostname|IP}/api/endeavour/catalog/application/database
```

Parameters

Parameter 1: `action`

Invoke the aggregate functions.

- **Value:** `aggregate`
- **Type:** System string. Required.

Data

Data 1: op

Use the `COUNT()` function to count the pk objects. For more information about the `COUNT()` function, follow the instructions in "COUNT()" on page 247. Name the output variable `count`.

- **Value:**

```
[
    {
        "operation":  "count",
        "fieldname":  "pk",
        "outputname": "count"
    }
]
```

- **Type:** Array. Required.

Example: Count databases

A Python snippet that is similar to the following example can be used to invoke an aggregate function that counts the total number of databases registered by IBM Spectrum Protect Plus. The number includes protected and unprotected databases.

```
_params = {"action": "aggregate"}

_data = f'''{{
    "op": [
        {{
            "operation":  "count",
            "fieldname":  "pk",
            "outputname": "count"
        }}
    ]
}}'''

requests.post('https://' + spp_ipv4
    + '/api/endeavour/catalog/application/database',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "results": [
        {
            "_id": null,
            "count": 100
        }
    ]
}
```

Counting protected virtual machines

You can count the virtual machines that are protected by IBM Spectrum Protect Plus.

Method and URI

To count protected virtual machines that are registered with IBM Spectrum Protect Plus, use a POST method and a URI:

```
POST    https://{hostname|IP}/api/endeavour/catalog/recovery/hypervisorvm
```

Parameters

Parameter 1: `action`

Invoke aggregate functions. For more information about aggregate functions, follow the instructions in “Aggregate functions” on page 246.

- **Value:** `aggregate`
- **Type:** System string. Required. Available in the web user interface.

Parameter 2: `filter`

Optionally, you may use a filter to specify the type of virtualized systems. If this parameter is not used, the request counts all protected virtual machines regardless of type. You can use the `filter` operation parameters that are described in “Filter” on page 243.

- **Example value:** To search protected virtual machines for only VMware instances, use the following filter parameter:

```
[
    {
         "property": "hypervisorType",
         "value":    "vmware",
         "op":       "="
    }
]
```

Use one of the following values to specify the virtualized system:

Virtualized system	Value
VMware vCenter	vmware
Hyper-V	hyperv
Amazon EC2	awsec2

- **Type:** Array. Available in the web user interface.

Data

Data 1: op

Use the `COUNT()` function for the pk values. For more information about the `COUNT()` function, follow the instructions in “COUNT()” on page 247.

- **Value:**

```
[
    {
        "operation":  "count",
        "fieldname":  "pk",
        "outputname": "count"
    }
]
```

- **Type:** Array. Required. Available in the web user interface.

Example 1: Count protected virtual machines

A Python snippet that is similar to the following example can be used to invoke an aggregate function that counts virtual machines that are protected by IBM Spectrum Protect Plus.

```
_params = {"action": "aggregate"}

_data = f'''{{
```

```
    "op": [
        {{
            "operation":  "count",
            "fieldname":  "pk",
            "outputname": "count"
        }}
    ]
}}'''

requests.post('https://' + spp_ipv4
    + '/api/endeavour/recovery/hypervisorvm',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "results": [
        {
            "_id": null,
            "count": 890
        }
    ]
}
```

Example 2: Count only VMware virtual machines

Add a filter to count only virtual machines that are hosted on VMware vCenter servers.

```
_params = {"action": "aggregate"}

_data = f'''{{
    "op": [
        {{
            "operation":  "count",
            "fieldname":  "pk",
            "outputname": "count"
        }}
    ],
    "filter": [
        {{
             "op":       "=",
             "property": "hypervisorType",
             "value":    "vmware"
        }}
    ]
}}'''

requests.post('https://' + spp_ipv4
    + '/api/endeavour/recovery/hypervisorvm',
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "results": [
        {
            "_id": null,
            "count": 700
        }
    ]
}
```

Counting protected databases

You can count the databases that are protected by IBM Spectrum Protect Plus.

Method and URI

To count protected databases that are registered with IBM Spectrum Protect Plus, use a POST method and a URI:

```
POST    https://{hostname|IP}/api/endeavour/catalog/recovery/applicationdatabase
```

Parameters

Parameter 1: `action`

Invoke aggregate functions. For more information about aggregate functions, follow the instructions in “Aggregate functions” on page 246.

- **Value:** `aggregate`
- **Type:** System string. Required. Available in the web user interface.

Data

Data 1: op

Use the `COUNT()` function for the pk values. For more information about the `COUNT()` function, follow the instructions in “COUNT()” on page 247.

- **Value:**

```
[
    {
        "operation":  "count",
        "fieldname":  "pk",
        "outputname": "count"
    }
]
```

- **Type:** Array. Required. Available in the web user interface.

Example: Count protected databases

A Python snippet that is similar to the following example can be used to invoke an aggregate function, `COUNT()`, which counts protected databases registered by IBM Spectrum Protect Plus.

```
_params = {"action": "aggregate"}

_data = f'''{{
    "op": [
        {{
            "operation":  "count",
            "fieldname":  "pk",
            "outputname": "count"
        }}
    ]
}}'''

requests.post('https://' + spp_ipv4
    + '/api/endeavour/catalog/recovery/applicationdatabase'
    headers={...}, params=_params, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
```

```
    "links": {...},
    "results": [
        {
            "_id": null,
            "count": 100
        }
    ]
}
```

Chapter 7. Accounts

You can manage users, roles, resource groups, and identities.

```
reference_user
reference_role
reference_resourcegroup
reference_identity
```

User

You can add users, get user information, and delete users.

Getting user information

You can get a list of users and information about each.

Method and URI

To get information about all users, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/security/user
```

To get information about a specific user, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/security/user/{userId}
```

Tip: To get a {userId} value based on the user name, follow the information in “Getting a {userId}” on page 238.

Parameters

None.

Data

None.

Example 1: Get information about all users

In this example, a Python snippet is used to get information about all users:

```
requests.get('https://' + spp_ipv4 + '/api/security/user',
     headers={...}, data="", verify=...)
```

This example shows a list of all users. The HTTP status of the response is 200 (OK). The response body below suggests that there are three users: Sarah (userId: 1005), sppadmin (userId: 1000) and Sales_Americas (userId: 1006), which is an LDAP group.

```
{
    "links": {...},
    "total": 3,
    "users": [
        {
            "links": {...},
            "name": "Sarah",
            "type": "NATIVE_USER",
```

```
            "typeDisplayName": "Native User",
            "tenantId": 1000,
            "loginCount": 0,
            "lastLogin": null,
            "failedLogin": 0,
            "lastFailedLogin": null,
            "lastPasswordUpdate": 1585007826754,
            "passwordAge": 0,
            "passwordExpiresAt": 0,
            "passwordExpired": false,
            "accountDisabled": false,
            "accountLocked": false,
            "rbacPath": "user:0/user:1005",
            "permissions": [
                {
                    "resourcePool": {
                        "links": {...},
                        "name": "BlueMachines - Dallas",
                        "description": "",
                        "total": 0,
                        "resources": [
                            {
                                "metadata": {
                                    "href_host": "",
                                    "href_path": "",
                                    "id": "1001",
                                    "name": "10.0.0.10",
                                    "typeTitle": "Hypervisor",
                                    "path": "Hypervisor:hypervisor/VMware:vmware
/vCenters:vcenters/10.0.0.10:1001"
                                },
                                "include": true,
                                "path": "root:0/hypervisor.all:0/hypervisor.type
:vmware/hypervisor:1001",
                                "displayName": "10.0.0.10"
                            },
                            {
                                "metadata": {
                                    "href_host": "",
                                    "href_path": "",
                                    "id": "all",
                                    "name": "All",
                                    "typeTitle": "All",
                                    "path": "Screen:screen/All:all"
                                },
                                "include": true,
                                "path": "root:0/screen:0",
                                "displayName": "All"
                            }
                        ],
                        "rbacPath": "resourcepool:0/resourcepool:1002",
                        "id": "1002"
                    },
                    "roles": [
                        {
                            "links": {...},
                            "name": "Self Service",
                            "type": "BUILTIN",
                            "description": null,
                            "displayName": "Self service",
                            "rbacPath": "role:0/role:992",
                            "id" "992",
                            "virtualresources": [
                                {
                                    "links": {...},
                                    "name": "Policy",
                                    "permissions": [
                                        {
                                            "name": "policy.view",
                                            "virtualResource": "Policy",
                                            "displayName": "View",
                                            "id": "policy.view"
                                        }
                                    ],
                                    "id": "Policy"
                                },
                                {
                                    "links": {...},
                                    "name": "Site",
                                    "permissions": [...],
                                    "id": "Site"
                                },
```

```
                                {...}, ..., {...}
                            ]
                        },
                        {
                            "links": {...},
                            "name": "Backup Only",
                            "type": "BUILTIN",
                            "description": null,
                            "displayName": "Backup only",
                            "rbacPath": "role:0/role:994",
                            "id": "994",
                            "virtualresources": [...]
                        }
                    ],
                    "permissionIds": [],
                    "roleIds": []
                }
            ],
            "id": "1005"
        },
        {
            "links": {...},
            "name": "sppadmin",
            "type": "NATIVE_USER",
            "typeDisplayName": "Native User",
            "tenantId": 1000,
            "loginCount": 1,
            "lastLogin": 1584468081598,
            "failedLogin": 0,
            "lastFailedLogin": 1584468072009,
            "lastPasswordUpdate": 1574191518133,
            "passwordAge": 0,
            "passwordExpiresAt": 0,
            "passwordExpired": false,
            "accountDisabled": false,
            "accountLocked": false,
            "rbacPath": "user:0/user:1000",
            "permissions": [
                {
                    "resourcePool": {
                        "links": {...},
                        "name": "All Resources",
                        "description": "Includes every resource in the system. ",
                        "total": 0,
                        "resources": [
                            {
                                "metadata": {},
                                "include": true,
                                "path": "root:0",
                                "displayName": "Unknown"
                            }
                        ],
                        "rbacPath": "resourcepool:0/resourcepool:1001",
                        "id": "1001"
                    },
                    "roles": [
                        {
                            "links": {...},
                            "name": "SUPERUSER",
                            "type": "BUILTIN",
                            "description": null,
                            "displayName": "SUPERUSER",
                            "rbacPath": "role:0/role:1000",
                            "id": "1000",
                            "virtualresources": []
                        }
                    ],
                    "permissionIds": [],
                    "roleIds": []
                }
            ],
            "id": "1000"
        },
        {
            "links": {...},
            "name": "Sales_Americas",
            "type": "LDAP_GROUP",
            "typeDisplayName": "LDAP Group",
            "tenantId": 1000,
            "loginCount": 0,
            "lastLogin": null,
            "failedLogin": 0,
```

```
            "lastFailedLogin": null,
            "lastPasswordUpdate": 1585045976570,
            "passwordAge": 0,
            "passwordExpiresAt": 0,
            "passwordExpired": false,
            "accountDisabled": false,
            "accountLocked": false,
            "rbacPath": "user:0/user:1006",
            "permissions": [...],
            "id": "1006"
        },
        {...}
    ]
}
```

Example 2: Get information about a specific user

The following Python snippet can be used to get information about a specific user: Sarah (`{userId}`: 1005).

```
user_id = "1005"

requests.get('https://' + spp_ipv4 + '/api/security/user/' + user_id,
     headers={...}, data="", verify=...)
```

```
{
    links": {...},
    "name": "Sarah",
    "type": "NATIVE_USER",
    "typeDisplayName": "Native User",
    "tenantId": 1000,
    "loginCount": 0,
    "lastLogin": null,
    "failedLogin": 0,
    "lastFailedLogin": null,
    "lastPasswordUpdate": 1585007826754,
    "passwordAge": 0,
    "passwordExpiresAt": 0,
    "passwordExpired": false,
    "accountDisabled": false,
    "accountLocked": false,
    "rbacPath": "user:0/user:1005",
    "permissions": [...],
    "personas": [],
    "id": "1007"
}
```

You can get a similar record if you use a `{userId}` for an LDAP group.

Adding an individual user

You can create an individual user. To register an LDAP group, follow the instructions in

"Adding an LDAP server" on page 132. To create users that are based on an LDAP group, follow the instructions in "Adding users that are based on an LDAP group" on page 196.

Method and URI

To add a user, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/api/security/user
```

Parameters

None.

Data

Data 1: `name`

The username of the new user. Do not use `root`, `admin`, or `test`.

- **Example value:** Sarah
- **Type:** String. Required. Available in the web user interface.

Data 2: `password`

The password of the new user. Use at least 8 characters.

- **Example value:** `MyPassw0rd!`
- **Type:** String. Required. Available in the web user interface.

Data 3: `type`

The type of user.

- **Value:** `NATIVE_USER`
- **Type:** System string. Available in the web user interface.

Data 4: `permissions`

Define permissions for the new user. Roles define what actions users can take and resource groups define what resources they can access.

- **Example value:** Use the following syntax to specify the roles and the resource groups for the user. You must specify the same roles for all resource groups.

```
[
    {
        "resourcePoolId": {resourcePoolId 1},
        "roleIds":        [
                              {roleId 1},
                              {roleId 2},
                              ...
                          ]
    }
    {
        "resourcePoolId": {resourcePoolId 2},
        "roleIds":        [
                              {roleId 1},
                              {roleId 2},
                              ...
                          ]
    },
    ...
]
```

Tip: To get a {roleId} value based on its name, follow the instructions in “Getting a {roleId} ” on page 238. To get the {resourcepoolId} value based on its name, follow the information “Getting a {resourcepoolId}” on page 239.

- **Type:** Array. Required. Available in the web user interface.

Example: Add a new individual user

Assume that you are an administrator of IBM Spectrum Protect Plus. A new member Sarah has joined your team. You want to create a new individual user for her.

Sarah must have permissions to view all hypervisor objects and some additional resources that are available in the appliance, such as the VMware instances. Sarah must have permissions to start backup jobs manually. But she must not start any other operations.

You can add the following user:

- **Username:** Sarah
- **Password:** `MyPassw0rd!`
- **Type:** An individual new user
- **Roles to assign:**
  - Backup Only (`{roleId}` 992)
  - Self Service (`{roleId}` 994)
- **Resource groups to access:**
  - BlueMachines - Dallas (`{resourcepoolId}` 1002)
  - Virtualized System All Resource Group (`{resourcepoolId}` 1000)

A Python snippet that is similar to the following example can be used to send a request to IBM Spectrum Protect Plus to add the new individual user:

```
_data = f'''{{
    "name":           "Sarah",
    "password":       "MyPassw0rd!",
    "type":           "NATIVE_USER",
    "sourceProvider": null,
    "sourceUser":     null,
    "sourceUserDN":   "",
    "permissions":    [
        {{
            "resourcePoolId": "1002",
            "roleIds":        ["992", "994"]
        }},
        {{
            "resourcePoolId": "1000",
            "roleIds":        ["992", "994"]
        }}
    ]
}}'''

requests.post('https://' + spp_ipv4 + '/api/security/user',
    headers={...}, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created). Review the response to ensure that the new user, Sarah, was added. The response body is similar to the JSON object for Sarah , as shown in “Getting user information” on page 191.

Adding users that are based on an LDAP group

You can create a group of users that are loaded from an LDAP group.

To register an LDAP group, follow the instructions in “Adding an LDAP server” on page 132. To create an individual user, follow the instructions in “Adding an individual user” on page 194.

Method and URI

To add a user, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/api/security/user
```

Parameters

None.

Data

Data 1: `name`

The CN of the LDAP group.

- **Example value:** `Sarah`
- **Type:** String. Required. Available in the web user interface.

Data 2: `type`

The type of user to add.

- **Value:** `LDAP_GROUP`
- **Type:** System string. Required. Available in the web user interface.

Data 3: `sourceProvider`

The `{ldapHref}` value of the LDAP server.

Tip: To get an {ldapHref} value based on the hostname or the IPv4 address of the LDAP server, follow the information in "Getting an {ldapHref}" on page 235.

- **Example value:** https://10.0.0.111/api/ldap/1012
- **Type:** System string. Required. Available in the web user interface.

Data 4: `sourceUser`

The `{ldapUserHref}` value of the LDAP server.

Tip: To get an {identityUserHref} value based on the identity name, follow the information in "Getting an {identityUserHref}" on page 232.

- **Example value:** https://10.0.0.111/api/ldap/1011/group/Sales_Americas
- **Type:** System string. Required. Available in the web user interface.

Data 5: `sourceUserDn`

The DN of the LDAP user. Use the syntax:

```
CN={CN},{groupRDN},{groupDN}
```

- **Example value:**

```
CN=Sales_Americas,OU=BM,OU=Security Groups,DC=BLUEMACHINES,DC=com.invalid
```

- **Type:** String. Required. Available in the web user interface.

Data 6: `permissions`

Define permissions for the new user. Roles define what actions users can take and resource groups define what resources they can access.

- **Example value:** Use the following syntax to specify the roles and the resource groups for the user. You must specify the same roles for all resource groups.

```
[
    {
        "resourcePoolId": {resourcePoolId 1},
        "roleIds":        [
                              {roleId 1},
                              {roleId 2},
```

```
                            ...
                        ]
    }
    {
        "resourcePoolId": {resourcePoolId 2},
        "roleIds":        [
                            {roleId 1},
                            {roleId 2},
                            ...
                        ]
    },
    ...
]
```

Tip: To get a {roleId} value based on its name, follow the information in “Getting a {roleId} ” on page 238. To get the {resourcepoolId} value based on its name, follow the information in “Getting a {resourcepoolId}” on page 239.

- **Type:** Array. Required. Available in the web user interface.

Example: Add users based on an LDAP group

Assume that you want to create users based on the following LDAP group in IBM Spectrum Protect Plus:

- **CN:** Sales_Americas
- **Type:** An LDAP group
- **{ldapHref}:** https://10.0.0.111/api/ldap/1011
- **{ldapUserHref}** https://10.0.0.111/api/ldap/1011/group/Sales_Americas
- **LDAP group search criteria:** `CN=Sales_Americas,OU=BM,OU=Security Groups`
  - **CN:** Sales_Americas
  - **Group RDN:** `OU=BM, OU=Security Groups`
  - **Base DN:** `DC=BLUEMACHINES,DC=com.invalid`
- **Roles to assign:** Backup Only (`{roleId}` 992)
- **Resource groups to access:** Virtualized System All Resource Group (`{resourcepoolId}`: 1000)

```
ldap_href = "https://10.0.0.111/api/ldap/1011"
ldap_user_href = "https://10.0.0.111/api/ldap/1011/group/Sales_Americas"

_data = f'''{{
    "name":           "Sales-Americas",
    "type":           "LDAP_GROUP",
    "sourceProvider": "{ldap_href}}",
    "sourceUser":     "{ldap_user_href}",
    "sourceUserDN":   "CN=Sales_Americas,OU=BM,OU=Security Groups,"
        + "DC=BLUEMACHINES,DC=com.invalid",
    "permissions":    [
        {{
            "resourcePoolId": "1000",
            "roleIds":        ["992"]
        }}
    ]
}}'''

requests.post('https://' + spp_ipv4 + '/api/security/user',
    headers={...}, data=_data, verify=...)
```

The request prompts a response with the HTTP status of 201 (Created). Review the response to ensure that the LDAP group, Sales_Americas, was added to the list of the users. The response body is similar to the JSON object for Sales_Americas, as shown in “Getting user information” on page 191.

Modifying settings of users

You can modify user settings, including the user name, roles, and assigned resource groups. Changing the password requires a different type of request. See “Changing a user password” on page 200.

Method and URI

To update the user {userId}, use a PUT method and a URI:

```
PUT     https://{hostname|IP}/api/security/user/{userId}
```

Tip: To get a {userId} value based on the user name, follow the information in “Getting a {userId}” on page 238.

Parameters

None.

Data

You can use the data key-value pairs that are described in “Adding an individual user” on page 194 or “Adding users that are based on an LDAP group” on page 196. To update the password of an individual user, follow the instructions in “Changing a user password” on page 200.

Example 1: Modify settings for an individual user

Assume that you have the following individual user:

- **Username:** Sarah (userId: 1005)
- **Roles:**
  - Backup Only
  - Self Service.
- **Resource groups:**
  - Virtualized System All Resource Group
  - BlueMachines - Dallas

You want to modify these user settings as shown:

- **Name:** Sarah Wiseman
- **Role:** SYSADMIN
- **Resource group:** All Resources

A Python snippet that is similar to the following example can be used to update the individual user settings:

```
user_id = "1005"    # Sarah (an individual user)

_data = f'''{{
    "name":         "Sarah Wiseman",
    "permissions": [
        {{
            "resourcePoolId": "1002",
            "roleIds":        ["1001"]
        }},
        {{
            "resourcePoolId": "1000",
            "roleIds":        ["1001"]
        }}
    ]
}}'''

requests.put('https://' + spp_ipv4 + '/api/security/user/' + user_id,
     headers={...}, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 200 (OK) and that the user is updated.

Example 2: Modify settings of a user that is based on an LDAP group

Assume that you added the user, Sales-Americas ({userId} 1006), to IBM Spectrum Protect Plus. The user is based on an LDAP group. You want to modify the settings as below:

- **Role:** SYSADMIN ({roleId} 1001)
- **Resource group:** All Resources ({resourcePoolId} 1001)

A Python snippet that is similar to the following example can be used to modify the settings of the LDAP group user:

```
user_id = "1006"    # Sales-Americas (an LDAP group)

_data = f'''{{
    "permissions": [
        {{
            "resourcePoolId": "1001",
            "roleIds":        ["1001"]
        }}
    ]
}}'''

requests.put('https://' + spp_ipv4 + '/api/security/user/' + user_id,
    headers={...}, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 200 (OK) and that the user is updated.

Changing a user password

You can change the password of an individual user. To modify other settings of users, follow the instructions in “Modifying settings of users” on page 198.

Tip: Passwords of users in an LDAP group are usually managed by the LDAP server.

Method and URI

To change the password, use a POST method and a URI:

```
POST    https://{hostname|IP}/api/security/user/{userId}
```

This POST request is idempotent.

Tip: To get a {userId} value based on the user name, follow the instructions in “Getting a {userId}” on page 238.

Parameters

Parameter 1: `action`

Start an action to change the user password.

- **Value:** `changePassword`
- **Type:** System string. Required. Available in the web user interface.

Data

Data 1: `newPassword`

A new password.

- **Example value:** `LGe;9BQd%L<4]cF-`
- String. Required. Available in the web user interface.

Example: Update the password of an individual user

Assume that you want to change the password of the user Sarah (userId: 1005) from `MyPassw0rd!` to `LGe;9BQd%L<4]cF-`. A Python snippet that is similar to the following example can be used to send a request to IBM Spectrum Protect Plus to update the password:

```
user_id = "1005"

_params = {"action": "changePassword"}

_data = f'''{{
    "newPassword": "LGe;9BQd%L<4]cF-"
}}'''

requests.put('https://' + spp_ipv4 + '/api/security/user/' + user_id,
    headers={...}, params=_params, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 204 (No Content) and that the user password is updated. There is no response body. Because this request is idempotent, you may send the same request and receive the same response without making any changes.

Deleting users

You can delete an individual user or remove an LDAP group from the list of users.

If you delete an LDAP server, the LDAP group will be automatically removed from the list of users. For details, follow the instructions in “Unregistering an LDAP server” on page 135.

Method and URI

To delete a user with a {userId}, the request's method and URI are similar to the following example:

```
DELETE  https://{hostname|IPv4}/api/security/user/{userId}
```

Tip: To get a {userId} value based on the user name, follow the information in “Getting a {userId}” on page 238.

Parameters

None.

Data

None.

Example

Assume that you want to delete user Sarah (`{userId}` 1005). Run the following command:

```
user_id = "1005"

requests.delete('https://' + spp_ipv4 + '/api/security/user/' + user_id,
    headers={...}, params="", data="", verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 204 (No Content) and that you no longer see the user Sarah.

To delete an LDAP group, send a similar request with the `{userId}` of the LDAP group.

Role

You can get information about roles. You can also add, edit, and delete roles.

Getting role information

You can get a list of roles and information about each.

Method and URI

To get information about all roles, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/security/role
```

To get information about a specific role, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/security/role/{roleId}
```

Tip: To get a {roleId} value based on its name, follow the information in “Getting a {roleId} ” on page 238.

Parameters

None.

Data

None.

Example 1: Get information about all roles

In this example, a Python snippet is used to get information about all roles:

```
requests.get('https://' + spp_ipv4 + '/api/security/role',
    headers={...}, data="", verify=...)
```

This example shows a list of roles. The HTTP status of the response is 200 (OK).


```
{
    "links": {...},
    "total": 10,
    "roles": [
        {
            "links": {...},
            "name": "VM Admin",
            "type": "BUILTIN",
            "description": null,
            "displayName": "VM administrator",
            "rbacPath": "role:0/role:997",
            "id": "997",
            "virtualresources": [
                {
                    "links": {...},
                    "name": "Policy",
                    "permissions": [
                        {
                            "name": "policy.create",
                            "virtualResource": "Policy",
                            "displayName": "Create",
                            "id": "policy.create"
                        },
                        {
                            "name": "policy.view",
                            "virtualResource": "Policy",
                            "displayName": "View",
                            "id": "policy.view"
                        },
                        {
                            "name": "policy.edit",
                            "virtualResource": "Policy",
                            "displayName": "Edit",
                            "id": "policy.edit"
```

```
                        },
                        {
                            "name": "policy.run",
                            "virtualResource": "Policy",
                            "displayName": "Run",
                            "id": "policy.run"
                        },
                        {
                            "name": "policy.delete",
                            "virtualResource": "Policy",
                            "displayName": "Delete",
                            "id": "policy.delete"
                        }
                    ],
                    "id": "Policy"
                },
                {
                    "links": {...},
                    "name": "Site",
                    "permissions": [
                        {
                            "name": "site.view",
                            "virtualResource": "Site",
                            "displayName": "View",
                            "id": "site.view"
                        }
                    ],
                    "id": "Site"
                },
                {...}, {...}, ..., {...}
            ]
        },
        {...}, {...}, ..., {...}
    ]
}
```

Example 2: Get information about a specific role

In this example, a Python snippet is used to get information about a specific role: VM Admin (`{roleId}` 1001).

```
role_id = "1001"

requests.get('https://' + spp_ipv4 + '/api/security/role/' + role_id,
    headers={...}, data="", verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "name": "VM Admin",
    "type": "BUILTIN",
    "description": null,
    "displayName": "VM administrator",
    "rbacPath": "role:0/role:997",
    "id": "997",
    "virtualresources": [
        {
            "links": {...},
            "name": "Policy",
            "permissions": [
                {
                    "name": "policy.create",
                    "virtualResource": "Policy",
                    "displayName": "Create",
                    "id": "policy.create"
                },
                {...}, {...}, ..., {...}
            ],
            "id": "Policy"
        },
        {
            "links": {...},
            "name": "Site",
            "permissions": [
                {
```

```
                        "name": "site.view",
                        "virtualResource": "Site",
                        "displayName": "View",
                        "id": "site.view"
                    }
                ],
                "id": "Site"
            },
            {...}, {...}, ..., {...}
        ]
    }
```

Creating a role

You can create a new role.

Method and URI

To create a new role, use a POST method and a URI:

```
POST    https://{hostname|IPv4}/api/security/role
```

Parameters

None.

Data

Data 1: `name`

The name of the role.

- **Example value:** Visitor
- **Type:** String. Required. Available in the web user interface.

Data 2: `description`

The description of the role.

- **Example value:** (None)
- **Type:** String.

Data 3: `permissionIds`

Specify pairs of a permission group and a permission item to enable. For example, to enable Application Server-View and Job-View, the `permissionIds` key should have the value `["appserver.view", "policy.view"]`. These permission items apply to both the web user interface and the REST API.

Permission group	Key	Available values
Databases / Cloud Management / Containers / File System	`application`	view
Application Server	`appserver`	register, view, edit, deregister
Certificate	`certificate`	create, view, edit, delete
Object Storage	`cloudprovider`	register, view, edit, deregister
Hypervisor	`hypervisor`	register, view, edit, deregister, options
Identity and Keys	`identity`	create, view, edit, delete
LDAP	`ldap`	register, view, edit, deregister

Permission group	Key	Available values
Log	`log`	view
Job	`policy`	create, view, edit, run, delete
VADP Proxy	`proxy`	register, view, edit, deregister
Report	`report`	create, view, edit, delete
Resource Group	`resourcepool`	create, view, edit, delete
Role	`role`	create, view, edit, delete
Script	`script`	upload, view, edit, delete
Script Server	`scriptserver`	register, view, edit, unregister
Site	`site`	create, view, edit, delete
SMTP	`smtp`	register, view, edit, deregister
Backup Storage	`storage`	register, view, edit, deregister
SLA Policy	`storageprofile`	create, view, edit, delete
User	`user`	create, view, edit, delete

In the REST API, you can toggle the permission to view and use the web user interface. This setting is not available in the **Role** pane of the web user interface.

Permission group	Key	Available values
Web User Interface	`screen`	view

Tip: Every user may have multiple roles. For example, assume that the user Sarah was associated with two system roles: VM Admin and Application Admin. In this case, she has all permissions items for both virtualized systems and application servers.

- **Example value:**

```
[
    "report.create",
    "report.view",
    "report.edit",
    "screen.view"
]
```

- **Type:** Array. Required. Available in the web user interface.

Example 1: Create a role that can view everything

Assume that you want to create a new role Visitor. You want to allow users that are associated with this role to view all permission groups, including Web User Interface. A Python snippet that is similar to the following example can be used to create the role:

```
_data = f'''{{
    "name":          "Visitor",
    "description":   "",
    "permissionIds": [
        "application.view",
        "appserver.view",
        "certificate.view",
        "cloudprovider.view",
        "hypervisor.view",
        "identity.view",
        "ldap.view",
```

```
            "log.view",
            "policy.view",
            "proxy.view",
            "report.view",
            "resourcepool.view",
            "role.view",
            "screen.view",
            "script.view",
            "scriptserver.view",
            "site.view",
            "smtp.view",
            "storage.view",
            "storageprofile.view",
            "user.view"
        ]
    }}'''

    requests.post('https://' + spp_ipv4 + '/api/security/role',
        headers={...}, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created). Review the response to ensure that the role Visitor was added. The response body is similar to the JSON object, as shown in **Example 2: Get information about a specific role**.

Example 2: Create a role that is permitted to only use the REST API

Assume that you want to create a role that has all permission items for Site, but the role must not use the web user interface.

As mentioned, revoking the permission to use the web user interface can be done from the REST API only. A Python snippet that is similar to the following example can be used to create such a role:

```
    _data = f'''{{
        "name":           "Screen Only",
        "description":    "",
        "permissionIds": [
            "site.create",
            "site.view",
            "site.edit",
            "site.delete"
        ]
    }}'''

    requests.post('https://' + spp_ipv4 + '/api/security/role',
        headers={...}, data=_data, verify=...)
```

If the user Sarah is only associated with this role, she can view, create, edit, and delete sites through the REST API, but she cannot see anything in the web user interface, including the **Site** page.

Editing the configuration of a role

You can edit the configuration of a role.

Method and URI

To edit the configuration of a role, use a PUT method with a URI:

```
PUT     https://{hostname|IPv4}/api/security/role/{roleId}
```

Tip: To get a {roleId} value based on the role name, follow the information in “Getting a {roleId} ” on page 238.

Parameters

None.

Data

For information about the available data key-value pairs, follow the instructions in

“Creating a role” on page 204

.

The update request will overwrite key-value pairs in data. Add the permission items that you want to keep in the `permissionIds` list. Any permission item that is not present in the list will be unselected and revoked.

Example: Edit the configuration of a role

Assume that you have a role with View permissions for all permission groups. The `{roleid}` is 1010. You want to modify the following values for this role:

- Rename it VisitorMod
- Keep the View permission for all permission groups
- Add the Run permission to Job
  - In the `permissionIds` list, this permission item is represented by `policy.run`.

A Python snippet that is similar to the following example can be used to update the role:

```
role_id = "1010"

_data = f'''{{
    "name":          "VisitorMod",
    "description":   "",
    "permissionIds": [
        "application.view",
        "appserver.view",
        ...,
        "policy.view",
        "policy.run",
        ...
    ]
}}'''

requests.put('https://' + spp_ipv4 + '/api/security/role/' + role_id,
    headers={...}, data=_data, verify=...)
```

After you run the Python snippet, ensure that you get a response with the HTTP status of 200 (OK) and that the role settings are updated. The response body is similar to the JSON object for Visitor, as shown in “Example 2: Get information about a specific role” on page 203.

Deleting a role

You can delete a custom role.

Method and URI

To delete a role with {roleId}, the request's method and URI are similar to the following example:

```
DELETE  https://{hostname|IPv4}/api/security/role/{roleId}
```

Tip: To get a {roleId} value based on its name, follow the information in “Getting a {roleId} ” on page 238.

Parameters

None.

Data

None.

Example: Delete a custom role

Assume that you have a role, Visitor ({roleId} 1002), and you want to delete it.

A Python snippet that is similar to the following example can be used to delete this role:

```
role_id = "1002"

requests.delete('https://' + spp_ipv4 + '/api/security/role/' + role_id,
    headers={...}, params="", data="", verify=...)
```

You will get a response with the HTTP status of 204 (No Content). Ensure you can no longer see the user Visitor.

Resource Group

You can complete operations for resource groups with the REST API, as described in the following section.

Getting resource group information

You can get a list of resource groups and information about each resource group.

Method and URI

To get information about all resource groups, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/security/resourcepool
```

To get information about a specific resource group, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/security/resourcepool/{resourcepoolId}
```

Tip: To get a {resourcepoolId} value based on its name, follow the information in “Getting a {resourcepoolId}” on page 239.

Parameters

None.

Data

None.

Example 1: Get information about all resource groups

In this example, a Python snippet is used to get information about all resource groups:

```
requests.get('https://' + spp_ipv4 + '/api/security/resourcepool',
    headers={...}, data="", verify=...)
```

The response body for the Python snippet displays a list of all resource groups. The HTTP status of the response is 200 (OK). There are four resource groups in the list.


```
{
    "links": {...},
    "total": 7,
    "resourcePools": [
        {
            "links": {...},
            "name": "Virtualized System All Resource Group",
            "description": "Includes all Hypervisor resources on the system",
            "total": 0,
            "resources": [
                {
                    "metadata": {
                        "path": "Hypervisor/All:0",
                        "name": "Hypervisor All",
                        "href_host": "http://localhost:8082"
                    },
                    "include": true,
                    "path": "root:0/hypervisor.all:0",
                    "displayName": "Hypervisor all"
                },
                {
                    "metadata": {
                        "path": "System Configuration/VADP Proxy/All:0",
                        "name": "VADP proxy All",
                        "href_host": "http://localhost:8082"
                    },
                    "include": true,
                    "path": "root:0/proxy:0/proxy.type:vmdkbackup.vadpproxy",
                    "displayName": "VADP proxy all"
                },
                {
                    "metadata": null,
                    "include": true,
                    "path": "root:0/policy:0/policytype:serviceprovider.recovery
    .file",
                    "displayName": "Unknown"
                },
                {
                    "metadata": {
                        "id": "1030",
                        "path": "Job/Hypervisor Recovery:1030",
                        "name": "Hypervisor Recovery",
                        "typeTitle": "Hypervisor Recovery",
                        "href_host": "http://localhost:8082",
                        "href_path": "/api/endeavour/policy/types/1030"
                    },
                    "include": true,
                    "path": "root:0/policy:0/policytype:serviceprovider.recovery
    .hypervisor",
                    "displayName": "Hypervisor recovery"
                },
                {
                    "metadata": {
                        "id": "1027",
                        "path": "Job/Hypervisor Inventory:1027",
                        "name": "Hypervisor Inventory",
                        "typeTitle": "Hypervisor Inventory",
                        "href_host": "http://localhost:8082",
                        "href_path": "/api/endeavour/policy/types/1027"
                    },
                    "include": true,
                    "path": "root:0/policy:0/policytype:serviceprovider.catalog.
    hypervisor",
                    "displayName": "Hypervisor inventory"
                },
                {
                    "metadata": {
                        "id": "1028",
                        "path": "Job/Hypervisor Backup:1028",
                        "name": "Hypervisor Backup",
                        "typeTitle": "Hypervisor Backup",
                        "href_host": "http://localhost:8082",
                        "href_path": "/api/endeavour/policy/types/1028"
                    },
                    "include": true,
                    "path": "root:0/policy:0/policytype:serviceprovider.protecti
```

```
on.hypervisor",
                    "displayName": "Hypervisor backup"
                },
                {
                    "metadata": {
                        "id": "allstorageprofiles",
                        "path": "SLA Policy/All:allstorageprofiles",
                        "name": "All SLA Policies",
                        "typeTitle": "All SLA Policies",
                        "href_host": "",
                        "href_path": "allstorageprofiles"
                    },
                    "include": true,
                    "path": "root:0/storageprofile:0",
                    "displayName": "All SLA policies"
                },
                {
                    "metadata": {
                        "id": "allscripts",
                        "path": "System Configuration/Scripts/All:allscripts",
                        "name": "All Scripts",
                        "typeTitle": "All Scripts",
                        "href_host": "",
                        "href_path": "allscripts"
                    },
                    "include": true,
                    "path": "root:0/script:0",
                    "displayName": "All scripts"
                },
                {
                    "metadata": {
                        "id": "allschedules",
                        "path": "Schedule/All:allschedules",
                        "name": "All Schedules",
                        "typeTitle": "All Schedules",
                        "href_host": "",
                        "href_path": "allschedules"
                    },
                    "include": true,
                    "path": "root:0/trigger:0",
                    "displayName": "All schedules"
                },
                {
                    "metadata": {
                        "id": null,
                        "path": "Report/VM Environment",
                        "name": "VM Environment",
                        "category": "",
                        "href_host": "http://localhost:8082",
                        "href_path": "/api/endeavour/report/categories/VM%20Envi
ronment"
                    },
                    "include": true,
                    "path": "root:0/report:0/reportcategory:VMEnvironmentCategor
yName",
                    "displayName": "VM environment"
                },
                {
                    "metadata": {
                        "id": "1014",
                        "path": "Report/Protection/VM SLA Policy Compliance",
                        "name": "VM SLA Policy Compliance",
                        "parentId": "",
                        "category": "Protection",
                        "href_host": "http://localhost:8082",
                        "href_path": "/api/endeavour/report/1014"
                    },
                    "include": true,
                    "path": "root:0/report:0/reportcategory:ProtectionCategoryNa
me/report:1014",
                    "displayName": "VM SLA policy compliance"
                },
                {
                    "metadata": {
                        "id": "1011",
                        "path": "Report/Protection/VM Backup History",
                        "name": "VM Backup History",
                        "parentId": "",
                        "category": "Protection",
                        "href_host": "http://localhost:8082",
                        "href_path": "/api/endeavour/report/1011"
                    },
```

```
                    "include": true,
                    "path": "root:0/report:0/reportcategory:ProtectionCategoryNa
me/report:1011",
                    "displayName": "VM backup history"
                },
                {
                    "metadata": {
                        "id": "1002",
                        "path": "Report/Protection/Protected VMs",
                        "name": "Protected VMs",
                        "parentId": "",
                        "category": "Protection",
                        "href_host": "http:/localhost/:8082",
                        "href_path": "/api/endeavour/report/1002"
                    },
                    "include": true,
                    "path": "root:0/report:0/reportcategory:ProtectionCategoryNa
me/report:1002",
                    "displayName": "Protected VMs"
                },
                {
                    "metadata": {
                        "id": "allkeys",
                        "path": "System Configuration/Keys/All:allkeys",
                        "name": "All Keys",
                        "href_host": "",
                        "href_path": "allkeys"
                    },
                    "include": true,
                    "path": "root:0/identity:0/identitytype:key:0",
                    "displayName": "All keys"
                },
                {
                    "metadata": {
                        "id": "allcertificates",
                        "path": "System Configuration/Certificates//All:allcerti
ficates",
                        "name": "All Certificates",
                        "href_host": "",
                        "href_path": "allcertificates"
                    },
                    "include": true,
                    "path": "root:0/certificate:0",
                    "displayName": "All certificates"
                },
                {
                    "metadata": {
                        "id": "allcloud",
                        "path": "System Configuration/Cloud//All:allcloud",
                        "name": "All Cloud",
                        "href_host": "",
                        "href_path": "allcloud"
                    },
                    "include": true,
                    "path": "root:0/cloud:0",
                    "displayName": "All cloud"
                },
                {
                    "metadata": {
                        "id": "allcredentials",
                        "path": "Accounts/Identity/All:allcredentials",
                        "name": "All Identities",
                        "href_host": "",
                        "href_path": "allcredentials"
                    },
                    "include": true,
                    "path": "root:0/identity:0/identitytype:credential:0",
                    "displayName": "All identities"
                },
                {
                    "metadata": {
                        "id": "9",
                        "name": "Dashboard",
                        "path": "Screen/Dashboard:1",
                        "href_host": "http:/localhost:8082",
                        "href_path": "/api/screen/9"
                    },
                    "include": true,
                    "path": "root:0/screen:0/screen:9",
                    "displayName": "Dashboard"
                },
                {
```

```
            "metadata": {
                "id": "10",
                "name": "File Restore",
                "path": "Screen/FileRestore:1",
                "href_host": "http:/localhost:8082",
                "href_path": "/api/screen/10"
            },
            "include": true,
            "path": "root:0/screen:0/screen:10",
            "displayName": "File restore"
        },
        {
            "metadata": {
                "id": "100",
                "name": "VMware Backup",
                "path": "Screen/VMware Backup:100",
                "href_host": "http:/localhost:8082",
                "href_path": "/api/screen/100"
            },
            "include": true,
            "path": "root:0/screen:0/screen:100",
            "displayName": "VMware Backup"
        },
        {
            "metadata": {
                "id": "101",
                "name": "VMware Restore",
                "path": "Screen/VMware Restore:101",
                "href_host": "http:/localhost:8082",
                "href_path": "/api/screen/101"
            },
            "include": true,
            "path": "root:0/screen:0/screen:101",
            "displayName": "VMware Restore"
        },
        {
            "metadata": {
                "id": "110",
                "name": "HyperV Backup",
                "path": "Screen/HyperV Backup:110",
                "href_host": "http:/localhost:8082",
                "href_path": "/api/screen/110"
            },
            "include": true,
            "path": "root:0/screen:0/screen:110",
            "displayName": "HyperV Backup"
        },
        {
            "metadata": {
                "id": "111",
                "name": "HyperV Restore",
                "path": "Screen/HyperV Restore:111",
                "href_host": "http:/localhost:8082",
                "href_path": "/api/screen/111"
            },
            "include": true,
            "path": "root:0/screen:0/screen:111",
            "displayName": "HyperV Restore"
        },
        {
            "metadata": {
                "id": "13",
                "name": "SLA Policy",
                "path": "Screen/Policy Overview:1",
                "href_host": "http:/localhost:8082",
                "href_path": "/api/screen/13"
            },
            "include": true,
            "path": "root:0/screen:0/screen:13",
            "displayName": "SLA policy"
        },
        {
            "metadata": {
                "id": "14",
                "name": "Backup Storage",
                "path": "Screen/BackupStorage:14",
                "href_host": "http:/localhost:8082",
                "href_path": "/api/screen/14"
            },
            "include": true,
            "path": "root:0/screen:0/screen:14",
            "displayName": "Backup storage"
```

```
                },
                {
                    "metadata": {
                        "id": "5",
                        "name": "Report",
                        "path": "Screen/Reports:5",
                        "href_host": "http:/localhost:8082",
                        "href_path": "/api/screen/5"
                    },
                    "include": true,
                    "path": "root:0/screen:0/screen:5",
                    "displayName": "Report"
                },
                {
                    "metadata": {
                        "id": "30",
                        "name": "Identity",
                        "path": "Screen/Identity:30",
                        "href_host": "http:/localhost:8082",
                        "href_path": "/api/screen/30"
                    },
                    "include": true,
                    "path": "root:0/screen:0/screen:30",
                    "displayName": "Identity"
                },
                {
                    "metadata": {
                        "id": "31",
                        "name": "Keys and Certificates",
                        "path": "Screen/Keys and Certificates:30",
                        "href_host": "http:/localhost:8082",
                        "href_path": "/api/screen/31"
                    },
                    "include": true,
                    "path": "root:0/screen:0/screen:31",
                    "displayName": "Keys and certificates"
                },
                {
                    "metadata": {
                        "id": "site",
                        "name": "Site",
                        "path": "System Configuration/Site/All:site",
                        "href_host": "http:/localhost:8082",
                        "href_path": ""
                    },
                    "include": true,
                    "path": "root:0/site:0",
                    "displayName": "Site"
                }
            ],
            "rbacPath": "resourcepool:0/resourcepool:1000",
            "id": "1000"
        },
        {
            "links": {...}
            },
            "name": "Database All Resource Group",
            ...,
            "rbacPath": "resourcepool:0/resourcepool:999",
            "id": "999"
        },
        {
            "links": {...},
            "name": "All Resources",
            "description": "Includes every resource in the system. ",
            ...,
            "rbacPath": "resourcepool:0/resourcepool:1001",
            "id": "1001"
        },
        {
            "links": {...},
            "name": "BlueMachines - Dallas",
            "description": "",
            "total": 0,
            "resources": [
                {
                    "metadata": {
                        "href_host": "",
                        "href_path": "",
                        "id": "1001",
                        "name": "10.0.0.10",
                        "typeTitle": "Hypervisor",
```

```
                        "path": "Hypervisor:hypervisor/VMware:vmware/vCenters:vc
    enters/10.0.0.10:1001"
                    },
                    "include": true,
                    "path": "root:0/hypervisor.all:0/hypervisor.type:vmware/hype
    rvisor:1001",
                    "displayName": "10.0.0.10"
                },
                {
                    "metadata": {
                        "href_host": "",
                        "href_path": "",
                        "id": "all",
                        "name": "All",
                        "typeTitle": "All",
                        "path": "Screen:screen/All:all"
                    },
                    "include": true,
                    "path": "root:0/screen:0",
                    "displayName": "All"
                }
            ],
            "rbacPath": "resourcepool:0/resourcepool:1002",
            "id": "1002"
        }
    ]
}
```


Example 2: Get information about a specific resource group

In this example, a Python snippet is used to get information about a specific resource group: BlueMachines - Dallas ({resourcepoolId} 1002).

```
resourcepool_id = "1002"

requests.get('https://' + spp_ipv4 + '/api/security/resourcepool/'
     + resourcepool_id,
     headers={...}, data="", verify=...)
```

The response will show information about the resource group. The HTTP status of the response is 200 (OK).

Identity

This section discusses how you can manage identities using the REST API. Some features in IBM Spectrum Protect Plus require credentials to access your resources. For example, assume that you want to protect a virtual machine that is hosted on VMware vCenter server. You will need the location and the credential of the vCenter server to grant IBM Spectrum Protect Plus access to the virtual machine. You can create an identity and associate it with the virtual machine. Similarly, you can create identities for Microsoft Windows Servers, IBM Db2 databases, an SMTP server, and so on.

Getting identity information

You can get a list of identities and information for each. Or get information about a specific identity.

Method and URI

To get information about all identities, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/identity/user
```

To get information about a specific identity, use a GET method and a URI:

```
GET     https://{hostname|IP}/api/identity/user/{identityUserId}
```

Tip: To get an {identityUserId} value based on its name, follow the information in “Getting an {identityUserId} ” on page 232.

Parameters

None.

Data

None.

Example 1: Get information about all identities

A Python snippet that is similar to the following example can be used to request a list of all identities and their information.

```
requests.get('https://' + spp_ipv4 + '/api/identity/user',
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "total": 3,
    "users": [
        {
            "links": {...},
            "resourceType": "user",
            "id": "2101",
            "name": "sarah-oracle1_10.4.4.1",
            "username": "sarah-oracle1",
            "type": "system",
            "ostype": null,
            "comment": null,
            "keyid": null,
            "realm": null,
            "rbacPath": "root:0/identity:0/identitytype:credential/identitycreden
   tial:2101",
            "password": null
        },
        {
            ...
            "id": "2102",
            "name": "sarah.wiseman@bluemachines.com.invalid_10.0.0.10",
            "username": "sarah.wiseman@bluemachines.com.invalid",
            ...
            "realm": "@bluemachines.com.invalid",
            "rbacPath": "root:0/identity:0/identitytype:credential/identitycreden
   tial:2102",
            ...
        },
        {
            ...
            "id": "2104",
            "name": ".\\Administrator_hyperv.dallas.bluemachines.com.invalid",
            "username": ".\\Administrator",
            ...
            "realm": "\\ADMINISTRATOR",
            "rbacPath": "root:0/identity:0/identitytype:credential/identitycreden
   tial:2104",
            ...
        },
        {
            ...
            "id": "2105",
            "name": "serveradmin_localhost",
            "username": "serveradmin",
            ...
            "realm": null,
            "rbacPath": "root:0/identity:0/identitytype:credential/identitycreden
   tial:2105",
            ...
        },
```

```
            {...}, {...}, ..., {...}
        ]
    }
```

Example 2: Get information about a specific identity

Assume that you added an identity, sarah-oracle1_10.4.4.1 (`{identityUserId}` 2101) to IBM Spectrum Protect Plus.

A Python snippet that is similar to the following example can be used to request information about this identity:

```
identity_user_id = "2101"

requests.get('https://' + spp_ipv4 + '/api/identity/user/' + identity_user_id,
    headers={...}, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 200 (OK).

```
{
    "links": {...},
    "resourceType": "user",
    "id": "2101",
    "name": "sarah-oracle1_10.4.4.1",
    "username": "sarah-oracle1",
    "type": "system",
    "ostype": null,
    "comment": null,
    "keyid": null,
    "realm": null,
    "rbacPath": "root:0/identity:0/identitytype:credential/identitycreden
01",
    "password": null
},
```

Adding an identity

You can add a new identity.

Method and URI

To add an identity, use a POST method with a URI:

```
POST    https://{hostname|IPv4}/api/identity
```

Parameters

None.

Data

Data 1: `name`

The name of the identity.

- **Example value:** AD-sarah.wiseman
- **Type:** String. Required. Available in the web user interface.

Data 2: `username`

The username of the identity.

- **Example value:** sarah.wiseman@bluemachines.com.invalid
- **Type:** String. Required. Available in the web user interface.

Data 3: `password`

The password of the identity.

- **Example value:** `@sw!9*Q7C.V<y-h"`
- **Type:** String. Required. Available in the web user interface.

Data 4: `ostype`

The operating system of the services in which the identity is used.

- **Value** Use `LINUX` for a credential to log in to a Linux system, or use `null` otherwise. For example, assume that you register an Oracle Database that is running on a Linux application server with IBM Spectrum Protect Plus. In this case, use `LINUX` for the application server, and use `null` for the Oracle Database.
- **Type:** System string or null. Required. Available in the web user interface.

Data 5: `comment`

An optional comment on the identity.

- **Example value:** Sarah's Active Directory credential.
- **Type:** String.

Example: Add an identity defined by Microsoft Active Directory

Assume that you want to register the following credential, which was created on Microsoft Active Directory:

- **Name:** AD-sarah.wiseman
- **Username:** sarah.wiseman@bluemachines.com.invalid
- **Password:** `@sw!9*Q7C.V<y-h"`
- **Comment:** Sarah's Active Directory credential

A Python snippet that is similar to the following example can be used to register the credential:

```
_data = f'''{{
    "name":     "AD-sarah.wiseman",
    "username": "sarah.wiseman@beluemachines.com.invalid",
    "password": "@sw!9*Q7C.V<y-h\\"",
    "ostype":   null,
    "comment":  "Sarah's Active Directory credential"
}}'''

requests.post('https://' + spp_ipv4 + '/api/identity/user',
    headers={...}, data=_data, verify=...)
```

Tip: To send a double quotation mark in the password value, type two backslashes in the Python f-string and the REST API receives a single backslash follow by the double quotation mark (\\").

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created). Review the response to ensure that the new identity, AD-sarah.wiseman, was created. The password value is represented by `null`.

```
{
    "links": {...},
    "resourceType": "user",
    "id": "2101",
    "name": "AD-sarah.wiseman",
    "username": "sarah.wiseman@beluemachines.com.invalid",
    "type": "system",
    "ostype": null,
    "comment": "Sarah's Active Directory credential",
    "keyid": null,
    "realm": "@BELUEMACHINES.COM.INVALID",
    "rbacPath": "root:0/identity:0/identitytype:credential/identitycredential:21
   33",
```

```
    "password": null
}
```

Editing an identity

You can edit an identity.

Method and URI

To update the identity with {identityUserId}, use a PUT method and a URI:

```
PUT     https://{hostname|IPv4}/api/identity/{identityUserId}
```

Parameters

None.

Data

You can use the data key-value pairs that were described in

"Adding an identity" on page 216.

Example: Update an identity defined by Microsoft Active Directory

Sarah has created an identity based on her own Microsoft Active Directory (AD) credential. Now she wants to use a shared account of the IT team instead:

- **Name:** BlueMachines-AD
- **Username:** AD-BlueMachines\Administrator
- **Password:** =q<gEbbN5G6$6xAh
- **Comment:** An Active Directory shared credential

A Python snippet that is similar to the following example can be used to update the identity settings. As you see in the password value, \ represents the backslash in Python's f-string and \ represents a backslash.

```
identity_user_id = "2101"

_data = f'''{{
    "name":     "BlueMachines-AD",
    "username": "AD-BlueMachines\\\\Administrator",
    "password": "=q<gEbbN5G6$6xAh",
    "comment":  "An Active Directory shared credential"
    }}'''

requests.put('https://' + spp_ipv4 + '/api/identity/user/' + identity_user_id,
    headers={...}, data=_data, verify=...)
```

The request prompts a response that is structured as shown, with the HTTP status of 201 (Created). Ensure the new identity AD-sarah.wiseman has been updated. The password value is represented by null.

```
{
    "links": {...},
    "resourceType": "user",
    "id": "2101",
    "name": "BlueMachines-AD",
    "username": "AD-BlueMachines\\Administrator",
    "type": "system",
    "ostype": null,
    "comment": "An Active Directory shared credential",
    "keyid": null,
    "realm": "\\ADMINISTRATOR",
```

```
    "rbacPath": "root:0/identity:0/identitytype:credential/identitycredential:21
   34",
    "password": null
}
```

Deleting an identity

You can delete an identity.

Method and URI

To delete an identity, use a DELETE method and a URI:

```
DELETE  https://{hostname|IPv4}/api/identity/user/{identityUserId}
```

Tip: To get an {identityUserId} value based on its name, follow the information in “Getting an {identityUserId} ” on page 232.

Parameters

None.

Data

None.

Example: Delete an identity

Assume that you have an identity AD-sarah.wiseman (`{identityUserId}` 2101).

A Python snippet that is similar to the following example can be used to delete the identity:

```
identity_user_id = "2101"    # AD-sarah.wiseman

requests.delete('https://' + spp_ipv4 + '/api/identity/user/'
    + identity_user_id,
    headers={...}, verify=...)
```

Ensure you get a response with the HTTP status of 204 (No Content) and that you no longer see the identity item AD-sarah.wiseman.

Chapter 8. Additional topics for the REST API

Learn about additional operations that you can complete with the REST API.

Collection resources

For collection resources, you can convert an object name to an ID. This operation might be necessary because the REST API often specifies objects by using their IDs instead of names.

For example, the following POST request can be used to apply an SLA Gold policy to the IBM Spectrum Protect Plus server with object name 10.0.0.100:

```
POST    https://10.0.0.100/api/endeavour/job/1000
```

In this POST request, you do not see the SLA policy name, Gold. Instead, you see its ID: 1000. Similarly, the SLA policy Silver usually has the ID of 1001 unless you delete and recreate the policy using the same name. Because you might not know the ID of an SLA policy, you need a function to convert object names into corresponding IDs.

The following Python snippet converts an SLA policy name into its ID. For more information about the `convert_sla()` function, follow the instructions in "Getting an {slapolicyId}" on page 241.

```
spp_url = "https://10.0.0.100"
spp_sessionid = "43b3b1b2cc434f4eb1b9df8c99e7866f"

sla_name = "Gold"

sla_id = convert_sla(spp_url, spp_sessionid, sla_name)
```

As shown in the previous example, the following information was required to log in to IBM Spectrum Protect Plus and obtain the SLA policy ID: the IBM Spectrum Protect Plus URL, the session ID, and the target object name, Gold.

In this example, the `convert_sla()` function will return the following ID for the Gold policy:

```
print(sla_id)
```

```
3001
```

Read the available information to learn how the converter functions can be used to convert a wide variety of object names into IDs.

Restriction: Within an object type, use unique object names. For example, all SLA policies should have unique names. Some third-party products support the use of identical names for multiple objects of the same type. For example, Hyper-V Server you can create two virtual machines having the same name in the same path. Avoid using duplicate names in an IBM Spectrum Protect Plus environment.

Application

For the application collection resource, you can use an object value to return another value for the same object.

Getting an {applicationName}

To specify the application, use one of the following system values:

Table 1. Application system values

Application	**{applicationName}**
VMware vCenter	`vmware`
Hyper-V	`hyperv`
Amazon EC2	`awsec2`
Microsoft 365	`office365`
Db2	`db2`
Microsoft Exchange Server	`exchange`
MongoDB	`mongo`
Oracle Database	`oracle`
Microsoft SQL Server	`sql`
IBM Spectrum Protect Plus	`catalog`

Getting an {applicationInstanceName}

IBM Spectrum Protect Plus assigns a name `{applicationInstanceName}` to each application instance on an application server.

The `[applicationInstanceName}` for instances of Db2, SQL Server, MongoDB, and Exchange Server has the following syntax:

```
{application server FQDN} / {application server name}
```

The `[applicationInstanceName}` for Oracle Database instances have the following syntax:

```
{application server host name} / {application server name}
```

Example 1: Assume that you installed a Db2 instance, db2americas, on a Linux server, whose FQDN is db2dallas1.bluemachines.com.invalid. The `{applicationName}` of this Db2 instance is structured as shown:

```
db2dallas1.bluemachines.com.invalid / db2americas
```

Example 2: Assume that you installed an Oracle Database instance, Oracle18cDBHome, on a Linux server, whose hostname is oracle-dallas1. The `{applicationName}` of this Oracle Database instance is structured as shown:

```
oracle-dallas1 / Oracle18cDBHome
```

Getting an {applicationInstanceId}

IBM Spectrum Protect Plus assigns an ID, `{applicationInstanceId}`, to each application instance on an application server.

Method and URI: To convert the value of an object for an application server, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/application/{applicationName}/instance
```

Path: Response body (JSON) > `instances` > `name` & `id`

Example: Assume that you registered the following Db2 application instance with IBM Spectrum Protect Plus:

- **{applicationName}:** db2
- **{applicationInstanceName}:** `db2dallas1.bluemachines.com.invalid / db2americas`

A Python snippet that is similar to the following example can be used to return its ID, `f794d5b0234ffa5271738a4bd9fbd905`:

```
application_name = "db2"
object_name = "db2dallas1.bluemachines.com.invalid / db2americas"

_response = requests.get('https://' + spp_ipv4 + '/api/application/'
    + application_name + '/instance,
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['instances']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
f794d5b0234ffa5271738a4bd9fbd905
```

Getting an {applicationDatabaseId}

IBM Spectrum Protect Plus assigns an ID, `{applicationDatabaseId}`, to each database on an application server.

Method and URI: To convert the value of an object for an application server, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/application/{applicationName}/instance
```

Path: Response body (JSON) > `instances` > `name` & `id`

Example: Assume that you registered the following Db2 application instance with IBM Spectrum Protect Plus:

- **{applicationName}:** db2
- **{applicationInstanceId}:** `f794d5b0234ffa5271738a4bd9fbd905`

A Python snippet that is similar to the following example can be used to return its ID, `b91891e551ee0ac0765472ac0e3ba041`:

```
application_name = "db2"
application_instance_id = "f794d5b0234ffa5271738a4bd9fbd905"
```

```
    # db2dallas1.bluemachines.com.invalid / db2americas

object_name = "db2dallas1.bluemachines.com.invalid / db2americas"

_response = requests.get('https://' + spp_ipv4 + '/api/application/'
    + application_name + '/instance,
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['instances']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['links']['self']

print(object_id)
```

```
b91891e551ee0ac0765472ac0e3ba041
```

Getting an {applicationDatabasegroupId} for SQL Server

IBM Spectrum Protect Plus assigns an ID, `{applicationDatabasegroupId}`, to each AG of SQL Servers.

Method and URI: To convert the value of an object for an application server, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/application/sql/databasegroup
```

Path: Response body (JSON) > `databasegroups` > `name` & `id`

Example: Assume that you registered the following SQL Server instances with IBM Spectrum Protect Plus:

- **Availability group name:** "bluemachines-ag"

A Python snippet that is similar to the following example can be used to return its ID, `372C4D0A-9069-439A-B356-7501A908270B`:

```
object_name = "bluemachines-ag"

_response = requests.get('https://' + spp_ipv4 + '/api/application/sql'
    + '/databasegroup',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['databasegroups']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['primaryKey']

print(object_id)
```

```
372C4D0A-9069-439A-B356-7501A908270B
```

Cloud

For the cloud collection resource, you can use an object value to return another value for the same object.

Getting a {cloudProviderInfoId}

IBM Spectrum Protect Plus assigns an ID `{cloudProviderInfoId}` to each object storage server or repository server.

Method and URI: To convert the name of a cloud object storage provider, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/cloud
```

Path: Response body (JSON) > `clouds` > `name` & `id`

Example: Assume that you have an object storage server that is named BlueMachines - IBM COS - Singapore. A Python snippet that is similar to the following example can be used to request the `{cloudProviderInfoId}` of the object storage server, 3:

```
object_name = "BlueMachines - IBM COS - Singapore"

_response = requests.get('https://' + spp_ipv4 + '/api/cloud',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)         # Convert to JSON

_response_json_object = _response_json['clouds']    # Get the specific object

for keys in _response_json_object:
    if keys['name'] == object_name:
        object_id = int(keys['id']))

print(object_id)
```

```
3
```

Getting a {cloudProviderInfoHref}

IBM Spectrum Protect Plus assigns a URL, `{cloudProviderInfoHref}`, to each object storage server or repository server.

Method and URI: To convert the value of an object for an identity, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/cloud
```

Path: Response body (JSON) > `clouds` > `name` & `href`.

Example: Assume that you have an object storage server that is named BlueMachines - IBM COS - Singapore. A Python snippet that is similar to the following example can be used to request the `{cloudProviderInfoHref}` of the object storage server, https://10.0.0.100/api/cloud/3:

```
object_name = "BlueMachines - IBM COS - Singapore"

_response = requests.get('https://' + spp_ipv4 + '/api/cloud',
```

```
    headers={...}, verify=...)

_response_json = json.loads(_response.text)         # Convert to JSON

_response_json_object = _response_json['clouds']    # Get the specific object

for keys in _response_json_object:
    if keys['name'] == object_name:
        object_href = keys['links']['self']['href']

print(object_href)
```

```
https://10.0.0.100/api/cloud/3
```

Global Preferences

For the Global Preferences, you can use the UI string of the preference to obtain the Global Preference object identifier (ID).

Getting an {globalpreferenceId}

IBM Spectrum Protect Plus assigns an ID, `{globalpreferenceId}`, to each Global Preference.

Method and URI: To convert the value of a string for a Global Preference to the ID, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/endeavour/preference
```

Path: Response body (JSON) > `preferences` > `name` & `id`.

Example: Assume that you want to obtain the ID for the Global Preference with the UI string "Enable SQL Server and Oracle databases restored in test mode eligible for backup" in IBM Spectrum Protect Plus. Copy the string from the UI being careful to avoid the space at the start and end of the string. A Python snippet that is similar to the following example can be used to return its `{globalpreferenceId}` value, `pref.application.common.catalog.iaMappedLunsSkipped`:

```
globalpref_name = "Enable SQL Server and Oracle databases restored in test mode eligible for
backup"

response = requests.get('https://' + spp_ipv4 + '/api/endeavour/preference',
    headers=..., verify=...)

response_json = json.loads(response.text)    # Convert to JSON

object_json = _response_json['preferences']

for keys in object_json:
    if keys['name'] == globalpref_name:
        globalpreferenceId = keys['id']

print(globalpreferenceId)
```

```
pref.application.common.catalog.iaMappedLunsSkipped
```

Hypervisor

For the hypervisor collection resource, you can use an object value to return another value for the same object.

Getting a {hypervisorHypervisorId}

IBM Spectrum Protect Plus assigns an ID, {hypervisorHypervisorId}, to each virtualized system server. The virtualized systems can be associated with vCenter Servers, Hyper-V servers, or Amazon EC2 accounts.

Method and URI: To convert the value of an object for a virtualized system, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/hypervisor/
```

Path: Response body (JSON) > `hypervisors` > `name` & `id`

Example: Assume that you added a vCenter Server instance, 10.0.0.10 (VMware-Dallas), to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{hypervisorHypervisorId}` value, `1001`:

```
object_name = "10.0.0.10"    # VMware-Dallas

_response = requests.get('https://' + spp_ipv4 + '/api/hypervisor/',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['hypervisors']

for keys in object_json:
    if keys['name'] == object_name
        object_id = keys['id']

print(object_id)
```

```
1001
```

Getting a {hypervisorDatacenterId}

In VMware vCenter, every vCenter Server has one or more datacenters. With IBM Spectrum Protect Plus, you can associate SLA policies with datacenters.

Method and URI: To convert the value of an object for a datacenter, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}
           /vmcontent
```

Path: Response body (JSON) > `contents` > `name` & `id`

Example: Assume that you added a vCenter Server, 10.0.0.10 (VMware-Dallas, `{hypervisorHypervisorId}` 1001), to IBM Spectrum Protect Plus. You have a datacenter, Dallas

- Datacenter2. A Python snippet that is similar to the following example can be used to return the datacenter's {hypervisorDatacenterId} value, 76edd18b1d3ce7865bbab693881ab53:

```
hypervisor_hypervisor_id = "1001"    # 10.0.0.10 (VMware-Dallas)
object_name = "Dallas-DataCenter2"

_response = requests.get('https://' + spp_ipv4 + '/api/hypervisor/'
    + hypervisor_hypervisor_id + '/vmcontent',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['contents']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
76edd18b1d3ce7865bbab693881ab53
```

Getting a {hypervisorFolderId} for a VMware environment

In an environment that is managed by VMware vCenter Server software, you can create multilevel folders in datacenters. IBM Spectrum Protect Plus assigns an ID, {hypervisorFolderId}, to each VMware folder and subfolder.

Method and URI: To get the ID of the folder in the datacenter folder (the root folder), use a GET method and a URI:

```
GET     https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}
           /datacenter/{hypervisorDatacenterId}/vmcontent
```

To get the ID of the subfolder in the folder or another subfolder with {hypervisorFolderId}, use a GET method and URI:

```
GET     https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}
           /folder/{hypervisorFolderId}/vmcontent
```

Path: Response body (JSON) > contents > name & id

Example: Assume that you added a VMware vCenter Server to IBM Spectrum Protect Plus. The vCenter Server name is 10.0.0.10 (VMware-Dallas) and the {hypervisorHypervisorId} value is 1001. You also have a datacenter that is named Dallas - Datacenter2 with a {hypervisorDatacenterId} value of 76edd18b1d3ce7865bbab693881ab53. In this datacenter, you added a folder, Production. A Python snippet that is similar to the following example can be used to return the {hypervisorFolderId} value, 6c5c1265cb91ca615ec16221fc0befe8:

```
object_name = "Production"

hypervisor_hypervisor_id = "1001"    # 10.0.0.10 (VMware-Dallas)
hypervisor_datacenter_id = "76edd18b1d3ce7865bbab693881ab53"
    # Dallas-DataCenter2
```

```
_response = requests.get('https://' + spp_ipv4 + '/api/hypervisor/'
    + hypervisor_hypervisor_id + '/datacenter/' + hypervisor_datacenter_id
    + '/vmcontent',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['contents']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
6c5c1265cb91ca615ec16221fc0befe8
```

Assume that you added a subfolder, online-services-subfolder, in the Production folder. A Python snippet that is similar to the following example can be used to return the {hypervisorFolderId} value of this subfolder, 51d0128ff2ece10133fc1e34acbbf86f:

```
object_name = "online-services-subfolder"

hypervisor_hypervisor_id = "1001"    # 10.0.0.10 (VMware-Dallas)
hypervisor_folder_id = "6c5c1265cb91ca615ec16221fc0befe8"    # Production

_response = requests.get('https://' + spp_ipv4 + '/api/hypervisor/'
    + hypervisor_hypervisor_id + '/folder/' + hypervisor_folder_id
    + '/vmcontent',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['contents']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
51d0128ff2ece10133fc1e34acbbf86f
```

In a VMware datacenter, you can create nested folders. Folders and subfolders within the hierarchy can have the same names, if the folders or subfolders with matching names have different folder paths. For example, assume that you have a VMware datacenter with folder New_York. You cannot create another folder using the same name, New_York, because these two folders share the same folder path. By contrast, you can create a subfolder, New_York, in the New_York folder because the two folders have different paths.

Getting a {hypervisorVmId}

Every virtual machine on VMware and Microsoft Hyper-V will have an ID, {hypervisorVmId}.

Method and URI: To get the ID of the folder in the datacenter folder (the root folder), use a GET method and a URI:

```
GET     https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}
```

```
            /datacenter/{hypervisorDatacenterId}/vmcontent
```

If the target VMware virtual machine is in a folder or a subfolder, use the following URI:

```
GET     https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}
            /folder/{hypervisorFolderId}/vmcontent
```

To get the ID of the virtual machine that is hosted by Hyper-V or Amazon EC2, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}
```

Path: Response body (JSON) > `contents` > `name` & `id`

Example: Assume that you have a virtual machine, Service-Dallas, in a subfolder, online-services-childfolder. The value of the `{hypervisorFolderId}` is `7a2ab40929457c84ed9c0e276a19f9ea`. The server name is 10.0.0.10 (VMware - Dallas) and the `{hypervisorHypervisorId}` is 1001. A Python snippet that is similar to the following example can be used to return the virtual machine's `{hypervisorVmId}` value, `ec44616651cda8b66fec4f7fd0daf554`:

```
object_name = "Service-Dallas"

hypervisor_hypervisor_id = "1001"    # 10.0.0.10 (VMware-Dallas)
hypervisor_folder_id = "7a2ab40929457c84ed9c0e276a19f9ea"
    # online-services-subfolder

_response = requests.get('https://' + spp_ipv4 + '/api/hypervisor/'
    + hypervisor_hypervisor_id
    + '/folder' + hypervisor_folder_id
    + '/vmcontent',
    headers=..., data=..., params=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['contents']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
ec44616651cda8b66fec4f7fd0daf554
```

Restriction: Do not use the same name for multiple virtual machines hosted on the same Hyper-V Server. With Microsoft Hyper-V, you can have multiple virtual machines sharing the same name in the same Hyper-V Server. Unlike VMware vCenter, Hyper-V does not support folders.

Getting a {hypervisorVdiskId} for a VMware environment

Every hard disk in a VMware virtual machine has an ID, `{hypervisorVdiskId}`.

Restriction: You can initiate operations for specific hard drives in a VMware vCenter environment, but not in a Microsoft Hyper-V environment or Amazon EC2 environment.

To get the ID of a virtual disk, use the GET method with a URI:

```
GET     https://{hostname|IP}/api/hypervisor/{hypervisorHypervisorId}
            /vm/{hypervisorVmId}/vmcontent
```

Path: Response body (JSON) > `contents` > `name` & `id`

Example: Assume that you want to get the `{hypervisorVdiskId}` of a hard disk, Hard disk 1, on a VMware virtual machine, Sales-Americas. The virtual machine has a `{hypervisorVmId}` of `ec44616651cda8b66fec4f7fd0daf554`. A Python snippet that is similar to the following example can be used to return the `{hypervisorVdiskId}` value, `e738a8a316cc11e8f7f84fa903e3dddc`:

```
object_name = "Hard disk 1"

hypervisor_hypervisor_id = "1001"    # 10.0.0.10 (VMware-Dallas)
hypervisor_vm_id = "ec44616651cda8b66fec4f7fd0daf554"    # Sales-Americas

_response = requests.get('https://' + spp_ipv4 + '/api/hypervisor/'
    + hypervisor_hypervisor_id
    + '/vm/' + hypervisor_vm_id
    + '/vmcontent',
    headers=..., data=..., params=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['contents']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
e738a8a316cc11e8f7f84fa903e3dddc
```

Getting a {hypervisorView}

To specify the type of view filter, use one of the following system values:

View filter	{hypervisor View}	VMware	Hyper-V	Amazon EC2
Datastore	`storageview`	x	x	
Tags	`tagview`	x		x
Tags and categories	`tagcategory`	x		
Host and clusters	`hostcontent`	x		

Identity

For the identity collection resource, you can use an object value to return another value for the same object.

Getting an {identityUserId}

IBM Spectrum Protect Plus assigns an ID, `{identityUserId}`, to each identity.

Method and URI: To convert the value of an object for an identity, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/identity/user
```

Path: Response body (JSON) > `users` > `name` & `id`

Example: Assume that you added an identity, `AD-sarah.wiseman`, to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{identityUserId}` value, 2133:

```
object_name = "AD-sarah.wiseman"

_response = requests.get('https://' + spp_ipv4 + '/api/identity/user',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['users']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
2133
```

Getting an {identityUserHref}

IBM Spectrum Protect Plus assigns a URL, `{identityUserHref}`, to each identity.

Method and URI: To convert the value of an object for an identity, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/identity/user
```

Path: Response body (JSON) > `keys` > `name` & `href`.

Example: Assume that you added an identity, `AD-sarah.wiseman`, to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{identityKeyHref}` value, https://10.0.0.100/api/identity/user/2133:

```
object_name = "AD-sarah.wiseman"

_response = requests.get('https://' + spp_ipv4 + '/api/identity/user',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['users']

for keys in object_json:
    if keys['name'] == object_name:
        object_href = keys['links']['self']['href']
```

```
print(object_href)
```

```
https://10.0.0.100/api/identity/user/2133
```

Getting an {identityKeyId}

IBM Spectrum Protect Plus assigns an ID, `{identityKeyId}`, to each access key or SSH key.

Method and URI: To convert the value of an object for an access key or SSH key, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/identity/key
```

Path: Response body > `keys` > `name` & `id`

Example: Assume that you added an access key, BlueMachines IBM COS Dallas Key, to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{identityKeyId}` value, 1003:

```
object_name = "BlueMachines IBM COS Dallas Key"

_response = requests.get('https://' + spp_ipv4 + '/api/identity/key',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['keys']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
1003
```

You can convert an SSH key name into its ID in a similar way.

Getting an {identityKeyHref}

IBM Spectrum Protect Plus assigns a URL, `{identityKeyHref}`, to each access key or SSH key.

Method and URI: To convert the value of an object for an access key or SSH key, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/identity/key
```

Path: Response body > `keys` > `name` & `links` > `self` > `href`.

Example: Assume that you added an access key, BlueMachines IBM COS Dallas Key, to IBM Spectrum Protect Plus. The Python code snippet that is similar to the following example can be used to return its `{identityKeyHref}` value, https://10.0.0.100/api/identity/key/1003:

```
object_name = "BlueMachines IBM COS Dallas Key"

_response = requests.get('https://' + spp_ipv4 + '/api/identity/key',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['keys']
```

```
for keys in object_json:
    if keys['name'] == object_name:
        object_href = keys['links']['self']['href']

print(object_href)
```

```
https://10.0.0.100/api/identity/key/1003
```

You can convert an SSH key name into its URL in a similar way.

Job

For the Job collection resource, you can use an object value to return another value for the same object.

Getting a {jobName}

IBM Spectrum Protect Plus assigns a name, `{jobName}`, to each scheduled job. Use one of the following policy names:

Scheduled job	Policy name
Maintenance	`Maintenance`
Storage server inventory	`Storage Server Inventory`
Application server inventory	`Application Server Inventory`
Hypervisor inventory	`Hypervisor Inventory`
Protection	`{applicationName}_{slaPolicyName}`

For more information about `{applicationName}`, follow the instructions in

"Getting an {applicationName}" on page 222.

Getting a {jobId}

IBM Spectrum Protect Plus assigns an ID, `{jobId}`, to each scheduled job.

Method and URI: To convert the value of an object for a scheduled job, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/endeavour/job
```

Path: Response body (JSON) > `jobs` > `policyName` & `id`.

Example: Assume that you added an SLA policy, Gold, to IBM Spectrum Protect Plus. Then, you assigned the policy to SQL Server instances. In this case, you must add the prefix "`sql_`" to the SLA policy name. A Python snippet that is similar to the following example can be used to return its `{jobId}` value, `1011`:

```
object_name = "sql_Gold"

_response = requests.get('https://' + spp_ipv4 + '/api/endeavour/job',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['jobs']

for keys in object_json:
    if keys['policyName'] == object_name
        object_id = keys['id']

print(object_id)
```

```
1011
```

LDAP

For the LDAP collection resource, you can use an object value to return another value for the same object.

Getting an {ldapId}

IBM Spectrum Protect Plus assigns an ID, `{ldapId}`, to each LDAP server.

Method and URI: To convert the value of an object for an LDAP server, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/ldap
```

Path: Response body (JSON) > `ldapServers` > `hostAddress` & `id`

Example: Assume that you added an LDAP server (IPv4 address 10.0.0.111) to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its ID, 1012:

```
object_name = "10.0.0.111"

_response = requests.get('https://' + spp_ipv4 + '/api/ldap',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['ldapServers']

for keys in object_json:
    if keys['hostAddress'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
 1012
```

Getting an {ldapHref}

IBM Spectrum Protect Plus assigns a URL `{ldapHref}` to the LDAP server that was registered with IBM Spectrum Protect Plus.

Method and URI: To convert the value of an object for an identity, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/ldap
```

Path: Response body (JSON) > `ldapServers` > `hostAddress` & `links` > `self` > `href`

Example: Assume that you added an LDAP server (IPv4 address 10.0.0.111), to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{ldapHref}` value, https://10.0.0.111/api/ldap/1012:

```
object_hostAddress = "10.0.0.111"

_response = requests.get('https://' + spp_ipv4 + '/api/ldap',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['ldapServers']

for keys in object_json:
    if keys['hostAddress'] == object_hostAddress:
        object_href = keys['links']['self']['href']
```

```
print(object_href)
```

```
https://10.0.0.111/api/ldap/1012/user/Sales-Americas
```

Getting an {ldapUserHref}

IBM Spectrum Protect Plus assigns a URL, `{ldapUserHref}`, to an LDAP server common name.

Method and URI: To convert the value of an object for an identity, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/ldap/{ldapId}/user
```

Path: Response body (JSON) > `users` > `cn` & `links` > `self` > `href`

Example: Assume that you added the LDAP server (`{ldapId}` 1012) and common name, Sales-Americas, to IBM Spectrum Protect Plus.

A Python snippet that is similar to the following example can be used to return its `{ldapUserHref}` value, https://10.0.0.111/api/ldap/1012/user/Sales-Americas:

```
ldapId = "1012"
ldapCn = "Sales-Americas"

_response = requests.get('https://' + spp_ipv4 + '/api/ldap',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['users']

for keys in object_json:
    if keys['cn'] == ldapCn:
        object_href = keys['links']['self']['href']

print(object_href)
```

```
https://10.0.0.111/api/ldap/1012/user/Sales-Americas
```

Report

For the report collection resource, you can use an object value to return another value for the same object.

Getting a {reportIdStr}

IBM Spectrum Protect Plus assigns an ID, `{reportIdStr}`, to each predefined report and custom report.

Method and URI: To convert the value of an object for a report, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/endeavour/report/
```

Path: Response body (JSON) > `reports` > `name` & `id`.

Example: Assume that you have a report, vSnapStorageUtilization-Dallas1. A Python snippet that is similar to the following example can be used to return its `{reportIdStr}`, 1013:

```
object_name = "vSnapStorageUtilization-Dallas1"

_response = requests.get('https://' + spp_ipv4 + '/api/report',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON
```

```
object_json = _response_json['reports']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
1013
```

Getting a {reportHref}

IBM Spectrum Protect Plus assigns a URL, {reportHref}, to each predefined report and custom report.

Method and URI: To convert the value of an object for a report, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/endeavour/report/
```

Path: Response body (JSON) > reports > name & links > self > href.

Example: Assume that you have a report, vSnapStorageUtilization-Dallas1. A Python snippet that is similar to the following example can be used to return its {reportHref} value, https://10.0.0.100/api/endeavour/report/1102:

```
object_name = "vSnapStorageUtilization-Dallas1"

_response = requests.get('https://' + spp_ipv4 + '/api/report',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['reports']   # Get the specific object

for keys in object_json:
    if keys['name'] == object_name:
        object_href = keys['links']['self']['href']

print(object_href)
```

```
https://10.0.0.100/api/endeavour/report/1102
```

Getting a {reportRbacpath}

The {reportRbacpath} is the system path of the role-based access control (RBAC) permissions that are assigned to each predefined report and custom report.

Method and URI: To convert the value of an object for a report, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/endeavour/report/
```

Path: Response body (JSON) > reports > name & rbacPath

Example: Assume that you have a report, vSnapStorageUtilization-Dallas1. A Python snippet that is similar to the following example can be used to return its {reportRbacpath} value.

```
object_name = "vSnapStorageUtilization-Dallas1"

_response = requests.get('https://' + spp_ipv4 + '/api/report',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['reports']   # Get the specific object
```

```
for keys in object_json:
    if keys['name'] == object_name
        object_value = keys['rbacPath']

print(object_value)
```

```
root:0/report:0/reportcategory:BackupStorageUtilizationCategoryName/report:1102
```

Security

For the security collection resource, you can use an object value to return another value for the same object.

Getting a {userId}

IBM Spectrum Protect Plus assigns an ID, `{userId}`, to each user.

Method and URI: To convert the value of an object for a user, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/security/user
```

Path: Response body (JSON) > `users` > `name` & `id`.

Example: Assume that you added a user, Sarah, to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return Sarah's `{userId}` value, 1001:

```
object_name = "Sarah"

_response = requests.get('https://' + spp_ipv4 + '/api/security/user',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['users']    # Get the specific object

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
1001
```

Getting a {roleId}

IBM Spectrum Protect Plus assigns an ID, `{roleId}`, to each role.

Method and URI: To convert the value of an object for a role, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/security/role
```

Path: Response body (JSON) > `roles` > `name` & `id`.

Example: Assume that you added a role, Visitor, to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{roleId}` value, 1002:

```
object_name = "Visitor"

_response = requests.get('https://' + spp_ipv4 + '/api/security/role',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON
```

```
_response_json_object = _response_json['roles']    # Get the specific object

for keys in _response_json_object:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
1002
```

Getting a {resourcepoolId}

IBM Spectrum Protect Plus assigns an ID, `{resourcepoolId}`, to each resource group.

Method and URI: To convert the value of an object for a resource group, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/security/resourcepool
```

Path: Response body (JSON) > `resourcepools` > `name` & `id`.

Example: Assume that you added a resource group, BlueMachines - Dallas, to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{resourcepoolId}` value, 1002:

```
object_name = "BlueMachines - Dallas"

_response = requests.get('https://' + spp_ipv4 + '/api/security/resourcepool',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['resourcepools']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
1002
```

Getting a {certificateId}

IBM Spectrum Protect Plus assigns an ID, `{certificateId}`, to each certificate.

Method and URI: To convert the value of an object for a certificate, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/security/certificate
```

Path: Response body (JSON) > `certificates` > `name` & `id`.

Example: Assume that you added a certificate, BlueMachines - Cert IBM Spectrum Protect, to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{certificateId}` value, 1286:

```
object_name = "BlueMachines - Cert IBM Spectrum Protect"

_response = requests.get('https://' + spp_ipv4 + '/api/security/certificate',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['certificates']
```

```
for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
1286
```

Getting a {certificateHref}

IBM Spectrum Protect Plus assigns a URL `{certificateHref}` to each certificate.

Method and URI: To convert the value of an object for a certificate, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/security/certificate
```

Path: Response body (JSON) > `certificates` > `name` & `links` > `self` > `href`.

Example: Assume that you added a certificate, "BlueMachines - Cert IBM Spectrum Protect", to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{certificateHref}` value:

```
object_name = "BlueMachines - Cert IBM Spectrum Protect"

_response = requests.get('https://' + spp_ipv4 + '/api/security/certificate',
    headers={...}, verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['certificates']

for keys in object_json:
    if keys['name'] == object_name:
        object_href = keys['links']['self']['href']

print(object_href)
```

```
https://10.0.0.100/api/security/certificate/1286
```

Site

For the site collection resource, you can use an object value to return another value for the same object.

Getting a {siteId}

IBM Spectrum Protect Plus assigns an ID, `{siteId}`, to each site.

Method and URI: To convert the value of an object for a site, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/site
```

Path: Response body (JSON) > `sites` > `name` & `id`.

Example: Assume that you added a site, Singapore, to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{siteId}` value, 3101:

```
object_name = "Singapore"
```

```
_response = requests.get('https://' + spp_ipv4 + '/api/site',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['sites']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
3101
```

SLA policy

For the SLA policy collection resource, you can use an object value to return another value for the same object.

Getting an {slapolicyId}

IBM Spectrum Protect Plus assigns an ID, `{slapolicyId}`, to each SLA policy.

Method and URI: To convert the value of an object for an SLA policy, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/spec/storageprofile
```

Path: Response body (JSON) > `slapolicies` > `name` & `id`.

Example: Assume that you added an SLA policy, Gold, to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{slapolicyId}` value, 2113:

```
object_name = "Diamond"

_response = requests.get('https://' + spp_ipv4 + '/api/spec/storageprofile',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['slapolicies']

for keys in object_json:
    if keys['name'] == object_name
        object_id = keys['id']

print(object_id)
```

```
2113
```

SMTP

For the SMTP collection resource, you can use an object value to return another value for the same object.

Getting an {smtpId}

IBM Spectrum Protect Plus assigns an ID, `{smtpId}`, to each SMTP server.

Method and URI: To convert the value of an object for an SMTP server, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/smtp
```

Path: Response body (JSON) > `smtps` > `hostAddress` & `id`

Example: Assume that you added an SMTP server (IPv4 address 10.0.0.121) to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{smtpId}` value, 1001:

```
object_name = "10.0.0.121"

_response = requests.get('https://' + spp_ipv4 + '/api/smtp',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['smtps']

for keys in object_json:
    if keys['hostAddress'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
1001
```

Storage

For the storage collection resource, you can use an object value to return another value for the same object.

Getting a {storageId}

IBM Spectrum Protect Plus assigns an ID, `{storageId}`, to each storage server.

Method and URI: To convert the value of an object for a storage server, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/storage
```

Path: Response body (JSON) > `storages` > `name` & `id`.

Example: Assume that you added a vSnap server, 10.0.1.1, to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its `{storageId}` value, 2101:

```
object_name = "10.0.1.1"

_response = requests.get('https://' + spp_ipv4 + '/api/storage',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['storages']

for keys in object_json:
    if keys['name'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
2101
```

VADP

For the VADP collection resource, you can use an object value to return another value for the same object.

Getting a {vadpId}

IBM Spectrum Protect Plus assigns an ID, `{vadpId}`, to each VADP proxy.

Method and URI: To convert the value of an object for a VADP proxy, use a GET method with a URI:

```
GET     https://{hostname|IP}/api/vadp
```

Path: Response body (JSON) > `vadps` > `ipAddr` & `id`

Example: Assume that you added a VADP proxy (IPv4 address 10.0.2.1) to IBM Spectrum Protect Plus. A Python snippet that is similar to the following example can be used to return its ID, 2102:

```
object_name = "10.0.2.1"

_response = requests.get('https://' + spp_ipv4 + '/api/vadp',
    headers=..., verify=...)

_response_json = json.loads(_response.text)    # Convert to JSON

object_json = _response_json['vadps']

for keys in object_json:
    if keys['ipAddr'] == object_name:
        object_id = keys['id']

print(object_id)
```

```
 2102
```

Operation parameters for JSON object output

Most responses from the REST API are nested JSON objects that contain multiple records in the JSON object format. A response can contain several records.

The REST API has optional operation parameters that transform a JSON object in a response. These functions are similar to those seen in the American National Standards Institute (ANSI) Structured Query Language (SQL), which uses the `WHERE` clause, the `ORDER BY` clause, and so on. You can include REST API operation parameters in your HTTP request and receive a response in JSON object format.

Filter

The `filter` parameter filters the JSON objects in a response by retrieving only the JSON objects that satisfy given criteria.

Parameter 1: `filter`

Filters JSON objects. The value is an array, which can include multiple criteria. You will get only the records that meet at least one of the given criteria.

- **Example:** Assume that you want to get records that meet one of the criteria:
  - **Criterion 1:** The value `acknowledged` is equal to `false`.

– **Criterion 2:** The value `jobId` is one of the items in the list: `["1005", "1014]`.

```
"filter": [
    {
        "property": "acknowledged",
        "value":    false,
        "op":       "=",
    },
    {
        "property": "jobId",
        "value":    ["1005", "1014"],
        "op":       "IN",
    }
]
```

This operation is equivalent to the following ANSI SQL query:

```
SELECT * FROM table WHERE (acknowledged = 0) OR (value IN ("1005", "1014"))
```

Parameter 1.1: `filter` > List > JSON object > `property`

The key name of the values to evaluate.

- **Type:** String. Required.

Parameter 1.2: `filter` > List > JSON object > `value`

The value to compare. The data format must match the `property` value to evaluate.

- **Type:** Various. Required.

Parameter 1.3: `filter` > List > JSON object > op

The operator to evaluate the criteria.

- **Value:** Use one of the following values:

Value	Description
=	The property is equal to the value.
<	The property is smaller than the value.
>	The property is greater than the value.
IN	The property exists in the set of the value.

- **Type:** System string. Required.

A Python snippet that is similar to the following example can be used to send a request to IBM Spectrum Protect Plus to get alert records that meet the specified criteria:

```
_params = {
    "sort": f'''[
        {{
            "property": "jobId",
            "direction": "ASC"
        }},
        {{
            "property": "first",
            "direction": "DESC"
        }}
    ]'''
}

_response = requests.get('https://' + spp_ipv4
    + '/api/endeavour/alert/message/download/csv',
    headers={...}, params=_params, verify=...)

print(_response.text)
```

Sort

The `sort` parameter sorts records by given object values.

Parameter 1: `sort`

Sorts records. The value is an array, where you can include multiple sort operators.

- **Example:** Assume that you want to sort records by the `jobId` values in ascending order, and then sort records by the `start` values in descending order. You can use the following parameter:

```
"sort": [
    {
        "property":  "jobId",
        "direction": "ASC"
    },
    {
        "property":  "start",
        "direction": "DESC"
    }
]
```

This operation is equivalent to the following ANSI SQL query:

```
SELECT *  FROM table  ORDER BY jobId ASC, start DESC
```

- **Type:** List.

Parameter 1.1: `sort` > List > JSON object > `property`

The value to sort.

- **Type:** String. Required.

Parameter 1.2: `sort` > List > JSON object > `direction`

The order (ascending or descending) in which to sort records.

- **Value:** Use one of the following values:

Value	Description
DESC	Sort values in descending order (default).
ASC	Sort values in ascending order.

  - **Type:** System string.

Pagination

Records in a REST API response are sorted by the `id` values in ascending order, unless you specify the sorting order by using the `sort` function. By default, the REST API splits the response body into sets of 100 records. You can see the next 100 records by specifying the `pageStartIndex` value. Or, you can increase the number of records in each response by specifying the `pageSize` value.

Parameter 1: `pageStartIndex`

The initial record number in the ordered records. The records are numbered 0, 1, 2, and so on. You can specify 0 or any positive integer for the pageStartIndex value.

- **Example:** Assume that you want to get records, starting with the 201st record. Each page contains 100 records by default. You can use the following parameter to get the records between the 201st and the 300th:

```
"pageStartIndex": 200
```

This operation is equivalent to the following ANSI SQL query:

```
SELECT *  FROM table  WHERE id > 200  LIMIT 100
```

- **Type:** Integer.

Parameter 2: `pageSize`

The number of records on each page. The default value is 100.

- **Example:** Assume that you want to get 500 records in a response. You can use the following parameter:

```
"pageSize": 500
```

This operation is equivalent to the following ANSI SQL query:

```
SELECT *  FROM table  LIMIT 500
```

- **Type:** Integer.

Aggregate functions

The REST API has aggregate functions. Each aggregate function can receive a set of values for each argument and return a single-value result for a set of input values. These functions are similar to those seen in ANSI SQL queries, which use the `COUNT()` function, the `MAX()` function, and so on.

General structure

You can include REST API aggregate functions in your HTTP request and receive a response in JSON object format. The HTTP requests must use the POST method, although the operations are idempotent.

Use the following parameter:

Parameter 1: `action`

Invoke the aggregate functions.

- **Value:** `aggregate`
- **Type:** System string. Required.

Data 1: op

Specify the aggregate function, the input values, and the variable name for the output value in the data value. The data value is a JSON array in which you can invoke multiple aggregate functions in a single request.

- **Example value:** The following JSON array shows the typical structure of a data value for invoking multiple aggregate functions:

```
[
    {
        "operation":  <aggregate function 1>,
        "fieldname":  <input values 1>,
        "outputname": <output variable name 1>
    },
    {
        "operation":  <aggregate function 2>,
        "fieldname":  <input values 2>,
        "outputname": <output variable name 2>
    },
    ...
]
```

- **Type:** Array.

Data 1.1: op > `operation`

Specify the aggregate function.

- **Value:** The following values are available:

Aggregate function	operation
COUNT()	`count`
MAX()	`max`
SUM()	`sum`

For more information about the aggregate functions, follow the instructions in `COUNT()`, `SUM()`, and `MAX()`.

- **Type:** System string. Required.

Data 1.2: op > `fieldname`

The names of the target objects that you want to use for arguments. You can specify one or more target objects.

- **Example value:** pk
- **Type:** String. Required.

Data 1.3: op > `outputname`

The key name for the output key-value pair.

- **Example value:** `countPrimaryKeys`
- **Type:** String. Required.

COUNT()

The `COUNT()` function returns the number of objects.

This operation is equivalent to the following ANSI SQL query:

```
SELECT COUNT(primary_key) FROM table
```

SUM()

The `SUM()` function returns the sum of a set of numbers.

This operation is equivalent to the following ANSI SQL query:

```
SELECT SUM(transfer_size) FROM table
```

MAX()

The `MAX()` function returns the maximum value in a set of values in a group.

This operation is equivalent to the following ANSI SQL query:

```
SELECT MAX(transfer_size) FROM table
```

GROUP clause

You can use the `group` clause to divide selected rows into groups such that the rows of each group have matching values in one or more columns or expressions. The `group` clause is equivalent to the GROUP BY clause in ANSI SQL.

Data 2: `group`

Specify a value to apply to the `group` clause.

- **Example value:** `["pk"]`
- **Type:** Array.

This operation is equivalent to the following ANSI SQL query:

```
SELECT COUNT(primary_key) FROM table GROUP BY pk
```

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows: © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at Legal copytrade.

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linear Tape-Open, LTO, and Ultrium are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Intel and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Red Hat®, OpenShift®, Ansible®, and Ceph® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

VMware, VMware vCenter Server, and VMware vSphere are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at IBM Privacy Policy and IBM’s Online Privacy Statement at IBM Product Privacy in the section entitled “Cookies, Web Beacons and Other Technologies,” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at IBM Product Privacy.

Glossary

A glossary is available with terms and definitions for the IBM Spectrum Protect family of products.

See the IBM Spectrum Protect glossary.

Index

SPP Reuse URLs

System requirements

Creating catalog backup job

Permissions

Backing up the IBM Spectrum Protect Plus application

Adding a key

Adding a certificate

Backing up and restoring VMware data

Virtual machine privileges

Creating a user account for an individual user

Configuring global preferences

Managing SLA policies

Assigning a static IP address

VADP system requirements

Adding an access key

Installing a vSnap server

Tiering

Installing a physical vSnap

Creating jobs and job schedules

Job types

Product messages

Product Number: 5737-F11