



PUBLIC
2023-05-21

SAP Service Manager

Content

- 1 Consuming Services in SAP BTP. 3**
- 1.1 SAP Service Manager. 3
 - What's New for SAP Service Manager 4
 - Working with SAP Service Manager. 7
- 1.2 Consuming SAP BTP Services from Various Environments. 73
 - Consuming SAP BTP Services from the Cloud Foundry Environment. 73
 - Consuming SAP BTP Services from the Neo Environment. 74
 - Consuming SAP BTP Services from the Kyma Environment. 74
 - Consuming Services in Other Environments Using the SAP Service Manager Instances. 75
 - Consuming SAP BTP Services from Kubernetes. 78

1 Consuming Services in SAP BTP

To use services in SAP BTP, you create a service instance, using either the SAP BTP cockpit, or the command-line tool of your runtime platform (for example SAP BTP Command Line Interface) and create bindings to retrieve access credentials.

In a Platform-as-a-Service (PaaS) environment, all external dependencies, such as databases, messaging and filing systems, are services. PaaS themselves are also considered services.

In SAP BTP, services are offered in a marketplace, from which users can create service instances on-demand. A service instance is a single instantiation of a service running on SAP BTP.

Service instances are created using a specific service plan. A service plan is a configuration variant of a service. For example, a database can be configured with various plan sizes; each is a different service plan.

To achieve the integration between a service and an application, you must deliver credentials to application. You can use bindings to generate credentials to communicate directly with a service instance.

Some runtime platforms, such as Cloud Foundry, allow binding service instances to your application to automatically deliver the credentials to the application.

[SAP Service Manager \[page 3\]](#)

SAP Service Manager service is the central registry for service brokers and platforms in SAP BTP.

[Consuming SAP BTP Services from Various Environments \[page 73\]](#)

Learn more about how to consume SAP BTP services from various runtime environments.

Related Information

[About Services](#)

[Discovery Center Service Catalog](#)

1.1 SAP Service Manager

SAP Service Manager service is the central registry for service brokers and platforms in SAP BTP.

It allows you to consume platform services in any connected runtime environment, track the creation and management of service instances, and share services and service instances between different environments.

SAP Service Manager supports services that implement the [Open Service Broker API](#) (OSBAPI), and can be consumed natively in OSBAPI-enabled environments such as Cloud Foundry and Kubernetes.

SAP Service Manager can be accessed via the SAP BTP cockpit, command line tool or API, and allows management of platforms, service brokers, service instances, and service bindings. It's tightly integrated with SAP BTP services, and enforces service access rules and quotas.

SAP Service Manager works with the following resources:

- **Platforms**
Platforms are Open Service Broker API-enabled software systems on which applications and services are hosted. With SAP Service Manager, you can register your platform and enable it to consume the SAP Business Technology Platform services from your native environment. This registration results in a returned set of credentials that are needed to deploy the SAP Service Manager agent.
- **Service Brokers**
A service broker acts as a broker between the SAP Service Manager and a platform's marketplace to advertise catalogs of service offerings and service plans. It also receives and processes the requests from the marketplace to provision, bind, unbind, and deprovision these offerings and plans.
- **Service Instances**
Service Instance is an instantiation of a service that makes the functionality of that service available for consumption.
- **Service Bindings**
Service bindings provide access details for an existing service instance. The access details can be found in the service binding credentials property, and typically include access URLs and credentials.
- **Service Plans**
Service plans represent sets of capabilities provided by a service offering. For example, database service offerings provide different plans for different database versions or sizes, while the SAP Service Manager plans offer different data access levels.
- **Service Offerings**
Service offerings represent an advertisement of the service that is supported by a service broker. Service offerings are related to one or more service plans.

Parent topic: [Consuming Services in SAP BTP \[page 3\]](#)

Related Information

[Consuming SAP BTP Services from Various Environments \[page 73\]](#)

1.1.1 What's New for SAP Service Manager

All Service Manager Related Releases for 2021

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
SAP Service Manager	Extension Suite Development Efficiency	Other	Secure X.509 certificate authentication with the Service Manager APIs	You can now issue tokens for accessing Service Manager APIs with the X.509 certificates. See Technical Access .		New	2021-08-26
SAP Service Manager	Extension Suite Development Efficiency	Other	SAP BTP service operator with X.509-Based Certificates	You can now install the SAP BTP service operator with X.509-based certificates to ensure secure communication with SAP Service Manager. See SAP BTP service operator Setup .		New	2021-08-26
SAP Service Manager	Extension Suite Development Efficiency	Other	Using SAP Service Manager to Create SaaS Provisioning Service Instances	You can now use SAP Service Manager to create environment-agnostic instances of the SaaS Provisioning service. This enables you to work with the SaaS Provisioning service from any environment of your choice.		New	2021-08-26
				<p>i Note</p> <p>For the Cloud Foundry and Kyma environments, use the environment-specific native tools to create SaaS Provisioning service instances.</p> <p>See Consuming Services in Other Environments Using the SAP Service Manager Instances.</p>			

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Service Manager	Extension Suite - Development Efficiency	Cloud Foundry	Re-branding of the SAP Cloud Service Management Service	SAP Cloud Service Management Service has been rebranded to SAP Service Manager. We've changed all our documentation, enablement materials, and tools accordingly.		Changed	2021-04-23
Service Manager	Extension Suite - Development Efficiency	Cloud Foundry	Update a Service Instance with btp CLI and SMCTL CLI	You can now update an existing service instance with the btp CLI or Service Manager Control (SMCTL) CLI. See update services/instance .		New	2021-04-22
Service Manager	Extension Suite - Development Efficiency	Cloud Foundry Other	SAP BTP Service Operator for Kubernetes	With the SAP Business Technology Platform (SAP BTP) service operator, you can provision and consume SAP BTP services in your Kubernetes cluster in a Kubernetes-native way. The SAP BTP service operator is based on the Kubernetes operator pattern so that you can consume SAP BTP services from within the cluster using Kubernetes native tools. See Kubernetes Consumption .		New	2021-02-25

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Service Manager	Extension Suite - Development	Cloud Foundry Other	Manage the Service	When working with sapcp CLI to manage the Service Management resources such as service instances, service bindings, service plans, and service offerings, it's now possible to use a resource's name instead of its ID.		New	2021-01-28
	Management Efficiency		Manage-ment Resources by Their Names	This change brings a more intuitive experience, as names are easier to remember, saving time needed to use other commands to get resources' IDs. See Service Management Commands for sapcp CLI .			
Service Manager	Extension Suite - Development Efficiency	Cloud Foundry Other	Service Management Control (SMCTL)	Service Manager Control (SMCTL) command-line now supports the explicit logout command and session timeout period of 30 minutes to better secure users' data and avoid malicious access. See SMCTL Session Termination .		New	2021-01-28

1.1.2 Working with SAP Service Manager

You can use the SAP Service Manager service capabilities from the SAP BTP cockpit to manage service instances and service bindings. Also, technical access is available via a command-line interface and REST APIs.

[Managing Services Using the SAP BTP Cockpit \[page 8\]](#)

Use the SAP BTP cockpit to view and manage all the services your subaccount is entitled to consume.

[Using the Service Manager Control \(SMCTL\) Command-Line Tool \[page 20\]](#)

Use the Service Manager Control (SMCTL) command-line tool to manage environments, brokers, service instances, and service bindings in SAP Service Manager environment.

[Working with SAP Service Manager APIs \[page 58\]](#)

The SAP Service Manager service API defines a centralized REST interface that allows the management of platforms, service brokers, service offerings, service plans, service instances, and service bindings.

[SAP Service Manager Commands for SAP BTP Command Line Interface \(btp CLI\) \[Feature Set B\] \[page 72\]](#)

Use the SAP BTP command line interface to work with the SAP Service Manager resources.

1.1.2.1 Managing Services Using the SAP BTP Cockpit

Use the SAP BTP cockpit to view and manage all the services your subaccount is entitled to consume.

1.1.2.1.1 Accessibility Features in SAP Service Manager

Optimize your experience when using SAP Service Manager in SAP BTP cockpit with features and settings that make the software work more efficiently.

i Note

Since SAP Service Manager features are shown in the SAP BTP cockpit, the same accessibility features that apply to the SAP BTP cockpit also apply to SAP Service Manager.

For more information about the accessibility features for SAP BTP cockpit, see [Accessibility Features in SAP BTP Cockpit](#).

For more information about screen reader support and keyboard shortcuts, see [Accessibility for End Users](#).

1.1.2.1.2 View and Manage Services from the Service Marketplace

View all offerings you're entitled to consume in your subaccount and create an instance of a service or an environment and a subscription to an application.

Overview

In the navigation bar of the SAP BTP cockpit, select [Services](#)→[Service Marketplace](#).

All resources your subaccount is entitled to consume are shown as tiles.

You can view details about a specific resource by selecting one of the tiles.

The details include a short description of a resource, links to the available documentation and support, the plans associated with the resource, its technical name, and the environments for which resource is available.

Creating Instances or Subscriptions

To consume any of the entitled offerings in your subaccount, you need to create an instance or a subscription.

Use an intuitive wizard that guides you step by step through the procedure.

There are two ways you can access this wizard on the Service Marketplace page:

- Click on **⋮** (*Create*) in the top-right corner of a selected tile. A wizard opens.
- If you're viewing details of a specific service, click on *Create* in the top-right corner.

For more information about how to create instances, see [Creating Service Instances \[page 9\]](#).

For more information about the subscriptions to multitenant applications, see [Subscribe to Multitenant Applications Using the Cockpit](#).

To learn about creating the instances of the Kyma environment, see [Create a Kyma Environment Instance from Service Marketplace](#).

1.1.2.1.3 Service Instances

View and manage service instances associated with your subaccount.

i Note

You can also manage your subscriptions using the cockpit. For more information, see [Subscribe to Multitenant Applications Using the Cockpit](#)

In the navigation bar of the SAP BTP cockpit, select *Services*→*Instances and Subscriptions*.

All created service instances in your account appear under a single table.

Selecting a specific instance opens a preview window with basic information, such as its ID, service, plan.

There are actions you can perform on a selected instance, depending on the environment to which your instance belongs.

1.1.2.1.3.1 Creating Service Instances

Create service instance to start consuming the service of your choice.

Related Links

[Creating Service Instances in Cloud Foundry \[page 10\]](#)

[Creating User-Provided Service Instances in Cloud Foundry Environment \[page 12\]](#)

[Creating Service Instances in the Kyma Environment \[page 12\]](#)

[Creating Service Instances in the Kubernetes Environment \[page 12\]](#)

[Creating Instances in Other Environments \[page 13\]](#)

1.1.2.1.3.1.1 Creating Service Instances in Cloud Foundry

The service instances that you create in your Cloud Foundry environment instance enable your Cloud Foundry apps to consume services natively from Cloud Foundry.

Prerequisites

If you're working in an enterprise account, you need to add quotas to the services you purchased in your subaccount before they appear in the service marketplace. Otherwise, only default free-of-charge services are listed. Quotas are automatically assigned to the resources available in trial accounts.

For more information, see [Configure Entitlements and Quotas for Subaccounts](#).

Procedure

You can create instances either directly from your subaccount or from a Cloud Foundry org.

Creating Service Instances from your Subaccount

1. In the SAP BTP cockpit, navigate to the subaccount in which you want to create a service instance.
For more information, see [Navigate to Orgs and Spaces](#).
2. In the navigation area, choose **Services > Instances and Subscriptions**.
All existing service instances in your subaccount appear under a single table.
3. Select **Create** in the top-right corner.
A wizard opens, offering you to configure your new instance:
 1. **Enter basic info for your instance.**
Choose a service for which you wish to create an instance from the dropdown list.
 2. Select one of the available service plans.
 3. Choose **Cloud Foundry** as a runtime environment.
 4. Select a space in your Cloud Foundry org to create the instance.
 5. Choose a name for your service instance, then choose **Next**.

→ Recommendation

Use a CLI-friendly name to enable the managing of your instances also with the CLI for SAP BTP.

CLI-friendly name is a short string (up to 32 characters) that only contains alphanumeric characters (A-Z, a-z), numbers from 0 to 9, periods, underscores, and hyphens.

It can't contain whitespace characters.

6. (Optional) Configure instance parameters

Specify a JSON file to upload or configure parameters manually in the JSON format, then choose [Next](#).

i Note

Some services support providing of additional configuration parameters during instance creation.

These parameters are provided in a JSON object with a schema that defines the properties for which to provide values, either in-line or as a file to upload.

For your convenience, we've prepopulated JSON schema with properties for those services that support this feature, so you only need to provide values.

For additional information about the supported configuration parameters, see the documentation of a particular service offering.

7. Review and verify the instance details.

Use the preview to verify the instance details, then choose [Create](#).

The new instance of the selected service is created in your Cloud Foundry space.

i Note

Creation of the instance can take a while.

To check the current creation status, see the **Status** column for the new instance in the [Service Instances](#) table.

Creating Service Instances from your Cloud Foundry Space

1. In SAP BTP cockpit, navigate to the subaccount in which you want to create a service instance.
2. In the navigation area, choose [Cloud Foundry](#) [Spaces](#).
All spaces in your subaccount appear.
3. Select the space in which you want to create a service instance.
4. In the navigation area, choose [Services](#) [Service Instances](#).
5. Select [Create](#) in the top-right corner and follow the instructions in the **New Instance or Subscription** wizard.
The new instance of the selected service is created in your Cloud Foundry space.

Labels

You can assign labels to service instances to make them searchable and organized within the subaccount according to various criteria.

For more information, see [Labels](#).


Procedure

1. In the navigation area, choose [Services](#) [Instances and Subscriptions](#).
2. Find the instance to which you want to assign labels under the [Instances](#) section of the page.
3. Select the actions (☰) menu and from the dropdown list, choose [Add Labels](#).

i Note

If there are already labels assigned to the service instance, the action becomes [Change Labels](#).

You can see the existing labels under the [Labels](#) column.



If you don't see the column, unhide it by clicking on  ([Configure Table Columns](#)).

1.1.2.1.3.1.2 Creating User-Provided Service Instances in Cloud Foundry Environment

User-provided service instances enable you to use services that aren't available in the service marketplace with your apps running in Cloud Foundry Environment.

Once created, user-provided service instances behave like service instances created through the marketplace, and you can apply all the relevant operations on them (update, bind, unbind, delete).

Procedure

1. In the SAP BTP, navigate to the subaccount in which you want to create a service instance.
2. In the navigation area, choose [Cloud Foundry](#) > [Spaces](#) . All spaces in your subaccount appear.
3. Select the space in which you want to create a user-provided service instance.
4. In the navigation area, choose [Services](#) > [Service Instances](#) .
5. Click on [Create](#) in the top-right corner, then from the dropdown list choose [Create User-Provided Service Instance](#) and follow the instructions in the wizard that opens.

For more information about the user-provided service instances, see [User-Provided Service Instances](#) .

1.1.2.1.3.1.3 Creating Service Instances in the Kyma Environment

To create and manage instances in Kyma, see [Using Services in the Kyma environment](#).

1.1.2.1.3.1.4 Creating Service Instances in the Kubernetes Environment

For more information about how to create instances in Kubernetes, see [Kubernetes Consumption](#).

Labels

You can assign labels to the service instances created in Kubernetes to make them searchable and organized within the subaccount according to various criteria.

Procedure

1. In the navigation area, choose ► [Services](#) ► [Instances and Subscriptions](#) ►.
2. Find the instance to which you want to assign labels under the *Instances* section of the page.
3. Select the actions (⋮) menu and from the dropdown list, choose [Add Labels](#).

i Note

If there are already labels assigned to the service instance, the action becomes [Change Labels](#).

You can see the existing labels under the [Labels](#) column.

If you do not see the column, unhide it by clicking on ⚙️ ([Configure Table Columns](#)).

For more information, see [Labels](#).

1.1.2.1.3.1.5 Creating Instances in Other Environments

Use the SAP BTP cockpit to create instances for environments in which the deployed applications don't bind natively to instances.

i Note

Service instances created for environments other than Cloud Foundry, Kyma, or Kubernetes environments have different ways of connecting to applications. For more information, see [Consuming Services in Other Environments Using the SAP Service Manager Instances \[page 75\]](#).

Procedure

1. Navigate to the subaccount in which you want to create a service instance.
2. In the navigation area, choose ► [Services](#) ► [Instances and Subscriptions](#) ►.
All existing service instances grouped by the environments appear.
3. Select [Create](#) in the top-right corner.
A **New Instance** wizard opens, offering you to configure your new instance:
 1. **Enter basic info for your instance**
Choose a service for which you wish to create an instance from the dropdown list.
 2. Select one of the available service plans.
 3. Choose [Other](#) as a runtime environment.
 4. Choose a name for your service instance, then choose [Next](#).

→ Recommendation

Use a CLI-friendly name to enable the managing of your instances also with the CLI for SAP Business Technology Platform.

CLI-friendly name is a short string (up to 32 characters) that only contains alphanumeric characters (A-Z, a-z), numbers from 0 to 9, periods, underscores, and hyphens.

It can't contain white spaces.

5. **(Optional) Configure instance parameters**

Specify a JSON file to upload or configure parameters manually in the JSON format, then choose [Next](#).

i Note

Some services support providing of additional configuration parameters during instance creation.

Pass these parameters in a valid JSON object that contains service-specific configuration parameters, provided either in-line or in a file.

For a list of supported configuration parameters, see the documentation of a particular service offering.

6. **Review and verify the instance details**

Use the preview to verify the instance details, then choose [Create](#).

The new instance is created.

i Note

Creation of the instance can take a while.

To check the current creation status, see the **Status** column of the new instance under the [Service Instance](#) table

Labels

You can assign labels to service instances to make them searchable and organized within the subaccount according to various criteria.


Procedure

1. In the navigation area, choose [Services](#) > [Instances and Subscriptions](#).
2. Find the instance to which you want to assign labels under the [Instances](#) section of the page.
3. Select the actions (☰) menu and from the dropdown list, choose [Add Labels](#).

i Note

If there are already labels assigned to the service instance, the action becomes [Change Labels](#).

You can see the existing labels under the [Labels](#) column.

If you do not see the column, unhide it by clicking on  ([Configure Table Columns](#)).

For more information, see [Labels](#).

1.1.2.1.3.2 Updating Service Instances

Update the name, plan, and configuration parameters of your service instance directly from your subaccount by using the SAP BTP cockpit.

Procedure

i Note

The procedure applies to service instances created in Cloud Foundry or Other Environments.

Service instances created in Kyma or Kubernetes can't be updated in the SAP BTP.

For more details, see [Using Services in the Kyma environment](#) and [Kubernetes Consumption](#).

1. In the navigation area of the SAP BTP cockpit, choose **Services > Instances and Subscriptions**. All existing service instances in your subaccount appear under a single table.
2. In the *Runtime Environments* column of the *Service Instances* table, find Cloud Foundry or Other Environments, and select the instance you want to update.

i Note

You update an instance by selecting the Actions (☰) menu, and then *Update Instance*.

You can find the Actions menu either at the end of each instance row of the *Service Instances* tables, or in the top-right section of the service instance details area that opens to the right if you click on an instance row.

3. In the **Update Instance** wizard that opens, follow the steps to update your instance. You can update the name of the instance and a plan.

i Note

You can update a subscription plan only if additional plans for the service whose instance you are creating are entitled to the subaccount in which you're updating the instance and if your service is eligible for plan updates.

1.1.2.1.3.3 Deleting Service Instances

Delete service instances that you created in your subaccount by using the SAP BTP cockpit.

Procedure

i Note

You can delete service instances created in Cloud Foundry or Other Environments.
Service instances created in Kyma or Kubernetes can't be deleted from the SAP BTP cockpit.
For more details, see [Using Services in the Kyma environment](#) and [Kubernetes Consumption](#).

1. In the navigation area, choose ► [Services](#) ► [Instances and Subscriptions](#) ►
All existing service instances in your subaccount appear under a single table.
2. You delete an instance by selecting the Actions (⋮) menu, and then [Delete Instance](#).
You can find the Actions menu either at the end of each instance's row, or in the top-right section of the service instance details area that opens to the right if you click on an instance row.

i Note

You can delete instances that are bound to applications and have active service keys or bindings.
If your instance has service keys or bindings, you will receive a message during the deletion informing you about them.

1.1.2.1.4 Service Bindings

Create service bindings to configure and deliver access credentials to your applications.

1.1.2.1.4.1 Binding Service Instances to Cloud Foundry Applications

Bind a service instance to your Cloud Foundry app to enable the automatic delivery of credentials needed to access the service instance from the application.

Prerequisites

- Deploy an application in the same space in which you plan to create the service instance. For more information, see [Deploy Business Applications in the Cloud Foundry Environment](#).
- Create a service instance. For more information, see [Creating Service Instances in Cloud Foundry \[page 10\]](#).

Procedure

1. In the navigation area, choose ► [Services](#) ► [Service Instances](#) ►. All existing service instances grouped by environments appear.
2. In the Cloud Foundry instances section, select the instance to which you want to bind the application.
3. In the service instance details section that opens to the right, select the Actions menu (☰) and then select [Bind Application](#).

i Note

If all the applications in your Cloud Foundry space have already been bound, or there are no applications, an error is returned.

4. In the **New Binding** wizard, choose the application to bind.
5. Provide the parameters for your binding by choosing one of the following options:
 - Upload a JSON file.
 - Configure in-line in a JSON format.

1.1.2.1.4.2 Service Keys

Unlike service bindings that are used to automatically generate credentials, service keys are used to manually configure credentials for users to consume marketplace services. Once you configure them for your service, local clients, apps in other spaces, or entities outside your deployment can access your service with these keys. We discuss service keys in the context of a Cloud Foundry environment.

Prerequisites

- You have an assigned [space developer](#) role. For more information, see [About Roles in the Cloud Foundry Environment](#).
- You've created a service instance. For more information, see [Creating Service Instances in Cloud Foundry \[page 10\]](#).

Procedure

i Note

You can create a service key for a Cloud Foundry instance either directly from your subaccount or from your Cloud Foundry org in your subaccount.

- In the first case, in navigation area, select ► [Services](#) ► [Instances and Subscriptions](#) ► after you've chosen your subaccount.
- In the second case, navigate first to ► [Cloud Foundry](#) ► [Spaces](#) ►, and then to ► [Services](#) ► [Instances and Subscriptions](#) ►.

1. Select a Cloud Foundry instance for which you want to create a service key.
2. In the service instance details area that opens to the right, select the Actions menu (☰).
3. Choose [Create Service Key](#).
4. In the **New Service Key** wizard, choose a name for your service key and provide configuration parameters either by uploading a JSON file or by configuring them in-line.

For the full list of the available configuration parameters, see the documentation of the particular service offering.

1.1.2.1.4.3 Creating Service Bindings in Kyma

To create and manage bindings in Kyma, see [Using Services in the Kyma environment](#).

1.1.2.1.4.4 Creating Service Bindings in Kubernetes

To create and manage bindings in Kubernetes, see [Kubernetes Consumption](#).

1.1.2.1.4.5 Creating Service Bindings in Other Environments

Create a service binding to obtain the access credentials to the service instance of a service you want to consume.

Procedure

1. In the SAP BTP cockpit, navigate to the subaccount in which you want to create a service instance.
2. Choose ► [Services](#) ► [Instances and Subscriptions](#) ▾.
All existing service instances in your subaccount appear under a single table.
3. Find the service instance for which you want to create a service binding.

→ Recommendation

To narrow down your search, look for **Other Environments** under the *Runtime Environment* column of the Service Instances table.

4. In the service instance details section that opens to the right, select the Actions (⋮) menu, and then [Create Binding](#).
5. In the **New Binding** wizard, choose a name for your binding and provide configuration parameters either by uploading a JSON file or by configuring them in-line.

You have created a new binding for the service instance.

i Note

To learn more about service instances and service bindings in other environments, see [Consuming Services in Other Environments Using the SAP Service Manager Instances \[page 75\]](#)

1.1.2.1.5 Working with Environment Instances

Create an instance of an environment to consume in your subaccount.

On a subaccount level, environments constitute the actual platform-as-a-service (PaaS) offering of SAP BTP that allows for the development and administration of business applications.

Each environment comes equipped with specific tools, technologies, and runtimes that you need to build applications. The availability of different environments allows for greater flexibility in your development process. For more information, see [Environments](#).

If you've created a multi-environment subaccount, you can use it as a single address to host various applications and offer diverse development options. For more information about creating multi-environment subaccounts, see [Create a Subaccount](#).

There are two ways to enable an environment for consumption in a subaccount, from the [Service Marketplace](#) and from the [Overview](#) pages of the SAP BTP cockpit. If you need more information about an environment and its available plans, we recommend that you use the [Service Marketplace](#). But if you are already familiar with the environment, use the [Overview](#) page.

Enabling Environments from the [Service Marketplace](#) Page

Environments are offered in the [Service Marketplace](#) of the SAP BTP cockpit together with other services.

To find them in your subaccount, from the navigation bar select **Services** > [Service Marketplace](#), and then from the dropdown menu of the [All Types](#) filter, select [Environments](#).

Choose the tile of the environment that you want to enable for consumption in your subaccount, and then in the top-right corner of the tile, select ***** (Create)** to open the wizard to manually configure your environment instance.

Currently, you can enable the Cloud Foundry or Kyma environment.

Enabling Environments from the [Subaccount Overview](#) Page

From the navigation bar, select [Overview](#).

Choose an environment you want to enable for consumption by selecting either [Enable Cloud Foundry](#) or [Enable Kyma](#).

This action opens a wizard to configure an environment instance.

i Note

Wizard that opens on this page is environment-specific, therefore its layout is slightly different from the wizard that opens when creating an environment instance on the [Service Marketplace](#) page.

For more information, see [Create a Kyma Environment instance from Service Marketplace](#).

Assigning Labels to Environment Instances

You can assign labels to environment instances to make them searchable and organized according to various criteria.

For more information, see [Labels](#).

i Note

For environment instances, these custom labels are user-defined and apply only to SAP BTP. They are not the same labels that might be defined by your environment broker.


Procedure

1. In the navigation area, choose ► [Services](#) ► [Instances and Subscriptions](#) ▾.
2. Find the instance to which you want to assign labels under the *Environment* section of the page.
3. Select the actions (⋮) menu and from the dropdown list, choose [Add Labels](#).

i Note

If there are already labels assigned to the service instance, the action becomes [Change Labels](#).

You can see the existing labels under the [Labels](#) column.

If you do not see the column, unhide it by clicking on  ([Configure Table Columns](#)).

1.1.2.2 Using the Service Manager Control (SMCTL) Command-Line Tool

Use the Service Manager Control (SMCTL) command-line tool to manage environments, brokers, service instances, and service bindings in SAP Service Manager environment.

[Installing the Service Manager Control \(SMCTL\) Command-Line Tool \[page 21\]](#)

Install the SMCTL Command-Line tool to start using this interface with the SAP Service Manager service resources.

[Subscribing to SAP Service Manager \[Feature Set A\] \[page 21\]](#)

Describes the procedure necessary for obtaining the credentials for SAP Service Manager so that you can access SAP BTP services in your subaccount:

[Assign the Subaccount Service Administrator Collection \[page 22\]](#)

Learn how to assign a new Subaccount Service Administrator role collection to a user of your choice.

[Logging in to SAP Service Manager \[page 23\]](#)

Describes the procedure to access the SAP Service Manager using the Service Manager Control (SMCTL) Command-Line tool.

[Service Manager Control \(SMCTL\) CLI Commands \[page 23\]](#)

See the list of all SMCTL commands available for the Service Manager resources.

1.1.2.2.1 Installing the Service Manager Control (SMCTL) Command-Line Tool

Install the SMCTL Command-Line tool to start using this interface with the SAP Service Manager service resources.

Procedure

1. Download the latest release, see <https://github.com/Peripli/service-manager-cli/releases/latest> . For older versions, see the [releases page](#) .
2. Extract the content of the downloaded archive (.zip for Windows or .tar.gz for Linux OS and Mac OS).
3. (Optional) Add the `smctl` executable path to the `<PATH>` environment variable.

1.1.2.2.2 Subscribing to SAP Service Manager [Feature Set A]

Describes the procedure necessary for obtaining the credentials for SAP Service Manager so that you can access SAP BTP services in your subaccount:

Context

i Note


The content in this section is only relevant for cloud management tools feature set A. For more information, see [Cloud Management Tools - Feature Set Overview](#).

If you're using cloud management tools feature set B, you don't need to subscribe to SAP Service Manager because the required role collections are automatically available to you without the whole subscription process.

Therefore, skip this whole section.

Procedure

1. Log in to SAP BTP cockpit and access your global account.
Access the cockpit using the link provided to you by your cloud operator or account executive and log in.
2. In the SAP BTP cockpit, navigate to your subaccount and select **Services** **Service Marketplace** . Find the **Service Manager** tile.

3. Select the icon  (*Create*) in the top-right corner of the tile.
4. A new wizard opens. Follow the steps to configure an instance of Service Manager.





1.1.2.2.3 Assign the Subaccount Service Administrator Collection

Learn how to assign a new Subaccount Service Administrator role collection to a user of your choice.

Context

You can assign users from default identity providers, and from custom identity providers, to a role collection. After having entered the user's user ID, choose the origin key of the identity provider and the e-mail address.

Procedure

1. Open the SAP BTP cockpit.
2. Go to your global account and subaccount (see).
3. Choose  [Security](#)  [Role Collections](#) .
4. Choose the *Subaccount Service Administrator*.
5. Go to the *Users* section and choose *Edit*.
6. Enter the user ID of the user that you want to assign to the role collection. If the user only exists in a connected identity provider, you must choose the identity provider and type in the e-mail address.
7. (Optional) To add more users, choose  (Add a user).
8. Save your changes.

You've now assigned this user to the role collection. The user has all of the authorizations of the role collection.

1.1.2.2.4 Logging in to SAP Service Manager

Describes the procedure to access the SAP Service Manager using the Service Manager Control (SMCTL) Command-Line tool.

Procedure

1. Execute the following command from your Service Manager Control (SMCTL) command-line interface:

```
smctl login -a https://service-manager.cfapps.<region domain> --param  
subdomain=<subdomain>
```

- To find the `region domain` for the SAP Service Manager, see [Regions and API Endpoints Available for the Cloud Foundry Environment](#).
 - The `subdomain` is the subaccount subdomain in which the Service Manager is registered. You can find it in the SAP Cloud cockpit [Overview](#) tab of your subaccount.
2. Input your user, with an assigned *Subaccount Service Administrator Role Collection*, and your password.

i Note

If two-factor authentication (2FA) is activated on the SAP Business Technology Platform landscape, then you need to append the passcode generated by the SAP Authenticator to your password.

For example, if your password is `Abcd` and the authenticator-generated passcode is `1234`, enter the password as **Abcd1234**.

1.1.2.2.5 Service Manager Control (SMCTL) CLI Commands

See the list of all SMCTL commands available for the Service Manager resources.

You can get help about each command directly in the SMCTL, by using the `--help` flag.

❖ Example

Run `smctl register-platform --help` to get information about how to use `smctl register-platform`.

Each SMCTL command has command-specific and global flags. Latter are used by all SMCTL commands for Service Manager.

The following commands are available:

- Log In and Log Out
 - [login \[page 25\]](#)
 - [logout \[page 28\]](#)
- Service Bindings

- [get-binding \[page 46\]](#)
- [list-bindings \[page 48\]](#)
- [bind \[page 45\]](#)
- [unbind \[page 49\]](#)
- Service Brokers
 - [register-broker \[page 35\]](#)
 - [update-broker \[page 36\]](#)
 - [list-brokers \[page 37\]](#)
 - [delete-broker \[page 38\]](#)
- Platforms
 - [register-platform \[page 29\]](#)
 - [update-platform \[page 31\]](#)
 - [list-platforms \[page 32\]](#)
 - [delete-platform \[page 33\]](#)
- Service Instances
 - [provision \[page 39\]](#)
 - [get-instance \[page 41\]](#)
 - [list-instances \[page 42\]](#)
 - [deprovision \[page 43\]](#)
- Service Offerings
 - [list-offerings \[page 50\]](#)
 - [marketplace \[page 53\]](#)
- Service Plans
 - [list-plans \[page 51\]](#)
- Miscellaneous
 - [status \[page 54\]](#)
 - [info \[page 55\]](#)
 - [version \[page 56\]](#)
 - [help \[page 57\]](#)

Related Information

[Using the Service Manager Control \(SMCTL\) Command-Line Tool \[page 20\]](#)

1.1.2.2.5.1 Login and Logout

1.1.2.2.5.1.1 login

Authenticates a user against a Service Manager instance.

Overview

```
smctl login
```

Usage

```
smctl login [flags]
```

Aliases

login, l

Parameters

Required		Global Flag
-a, --url	Base URL for SAP Service Manager.	No

Required		Global Flag
--param	Use it to add more parameters. Parameters are specified as key=value pairs.	No

i Note

- You must have an assigned *Subaccount Service Administrator* role. For more information, see [Assign the Subaccount Service Administrator Collection \[page 22\]](#).
- The `subdomain` parameter is mandatory for login. Its value is the *Subdomain* of your subaccount, found in the SAP BTP cockpit on the subaccount [Overview](#) page.

Optional		Global Flag
-h, --help	Help for the <code>login</code> command.	No
-p, --password	User password	No
--skip-ssl-validation	Skip verification of the OAuth endpoint. Not recommended.	No
-u, --user	User ID	No
--config	Set the path for the <code>smctl config.json</code> file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use the verbose mode.	Yes
--auth-flow	Specify the authorization flow. You have the following two options: <ul style="list-style-type: none"> <code>password</code> <code>client-credentials</code> If not specified, <code>password</code> flow is used.	No

Optional		Global Flag
<code>--client-id</code>	<code>--client-secret</code>	<p>You specify these parameters if you chose the client-credentials authorization flow and if you created the binding object with the default credentials type.</p> <p>See the example of a binding object created with the default credentials type: Create a SAP Service Manager Instance and Binding [page 61].</p>
<code>--cert</code>	<code>--key</code>	<p>You specify these parameters if you chose the client-credentials authorization flow and if you created the binding object with the X.509 credentials type.</p> <p>Their values are paths to files that contain public and private keys respectively that were generated upon the creation of the binding object with X.509 credentials type.</p> <p>See the example of a binding object created with X.509 credentials type: Create a SAP Service Manager Instance and Binding [page 61].</p>

Password Flow Login Examples

```
> smctl login -a https://service-management-url.com --param
subdomain=<my_subaccount_subdomain_name>
User: user # entering username
Password: # entering password (password visibility is disabled)
Logged in successfully.
```

```
> smctl login -a https://service-management-url.com --param
subdomain=<my_subaccount_subdomain_name> -u user -p pass
Logged in successfully.
```

Client Credentials Login Examples

Client Credentials with the Default Type

```
> smctl login -a https://service-manager-url.com --param
subdomain=<my_subaccount_subdomain_name> --auth-flow=client-credentials --client-
id=id --client-secret=secret
    Logged in successfully.
```

Client Credentials with the X.509 Type

```
smctl login -a https://service-manager-url.com --param
subdomain=<my_subaccount_subdomain_name> --auth-flow=client-credentials --client-
id=id --cert=cert.pem --key=key.pem
    Logged in successfully.
```

i Note

SAP Service Manager Control (SMCTL) command-line also supports the explicit logout command and session timeout period of 30 minutes to better secure users' data and avoid malicious access.

Related Information

[Logging in to SAP Service Manager \[page 23\]](#)

1.1.2.2.5.1.2 logout

Overview

smctl logout

Logs the user out and deletes the active client access token.

Usage

```
smctl logout [flags]
```

Aliases

logout, v

Parameters

Optional		Global Flag
-h, --help	Help for the <code>logout</code> command.	No
-v, --verbose	Use the verbose mode.	Yes
--config	Use this parameter to specify which configuration file to remove during the logout. If no path is provided, a default path <code>\$HOME/.sm/config.json</code> is used.	Yes

Example 1

```
> smctl logout
You have successfully logged out.
```

1.1.2.2.5.2 Platforms

1.1.2.2.5.2.1 register-platform

Overview

`smctl register-platform`

Registers a platform in the SAP Service Manager instance.


```
PlhcjHaWLiZfu0T0/8LzM= cQ6Uq1v1x1AT+eBlzkuFLUZBJJlMt2KN6w2eQH/  
MdjQsYRdjFCZKwKHz0DLJCvaZHa/061yggmZ5nQabxtXpq/  
p90xccc4yLEhDZBrFzhYqc8c2l45NuNlZfwBsL3eq/o2sEddu0zz10K1M7JnBcztiTM7D0eycS7uWF02/  
K0PU=
```

1.1.2.2.5.2.2 update-platform

Overview

smctl update-platform

Updates a platform with the SAP Service Manager instance.

Usage

```
smctl update-platform [name] <json_platform> [flags]
```

Input Example

```
smctl update-platform platform '{"name": "new-name", "description": "new-  
description", "type": "new-type"}'
```

Aliases

update-platform, up

Parameters

Optional		Global Flag
-h, --help	Help for update-platform command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No

Optional		Global Flag
<code>--config</code>	Set the path for the smctl config.json file (default is <code>\$HOME/.sm/config.json</code>).	Yes
<code>--regenerate-credentials</code>	Whether to regenerate credentials for credentials rotation.	
<div style="border: 1px solid #ccc; padding: 5px; background-color: #f9f9f9;"> <p>i Note</p> <p>Once new credentials are generated, old credentials can no longer be used.</p> </div>		
<code>-v, --verbose</code>	Use verbose mode.	Yes

Output Example

```
> smctl update-platform sample-platform '{"description": "Sample platform instance"}'
ID              Name              Type
Description     Created          Updated
-----
6352fca0-c252-43ab-9cb3-d23613749b59 sample-platform sample Sample platform
instance 2018-07-18T07:06:40Z 2018-07-18T07:09:48Z
```

1.1.2.2.5.2.3 list-platforms

Overview

smctl list-platforms

Lists all platforms registered in the SAP Service Manager instance.

Usage

```
smctl list-platforms [flags]
```


Aliases

list-platforms, lp

Parameters

Optional		Global Flag
-h, --help	Help for <code>list-platforms</code> command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No
--config	Set the path for the <code>smctl config.json</code> file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl list-platforms
One platform registered.
ID                               Name           Type           Description
Created                         Updated
-----
6352fca0-c252-43ab-9cb3-d23613749b59 sample-platform sample Sample platform
2018-07-18T07:06:40Z 2018-07-18T07:06:40Z
```

1.1.2.2.5.2.4 delete-platform

Overview

`smctl delete-platform`

Deletes one or more platforms registered in the SAP Service Manager instance.

Usage

```
smctl delete-platform [name] <name2 <name3> ... <nameN>> [flags]
```

Aliases

delete-platform, dp

Parameters

Optional		Global Flag
-h, --help	Help for <code>delete-platform</code> command.	No
--cascade-delete	Deletes all the platforms associated with the subaccount in a single command execution.	No
--config	Set the path for the <code>smctl config.json</code> file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl delete-platform sample-platform  
Platform with name: sample-platform successfully deleted
```

```
> smctl delete-platform sample-platform --cascade-delete  
Cascade delete successfully scheduled for platform: sample-platform. To see  
status of the operation use:  
smctl status /v1/platforms/baea022b-64c0-43d4-a9b0-e1ae64af51cd/operations/  
f8ca64af-e889-4a45-ad41-f1baa2e427c2
```

1.1.2.2.5.3 Brokers

1.1.2.2.5.3.1 register-broker

Overview

`smctl register-broker`

Registers a broker in the SAP Service Manager instance.

Usage

```
smctl register-broker [name] [url] <description> [flags]
```

Aliases

register-broker, rb

Parameters

Required		Global Flag
-b, --basic	Sets the username and password for basic authentication. Format is <username:password> .	No
Optional		Global Flag
-h, --help	Help for <code>register-broker</code> command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text.	No
--config	Set the path for the <code>smctl config.json</code> file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl register-broker sample-broker-1 https://demobroker.domain.com/ "Service
broker providing some valuable services" -b user:pass
ID                               Name
URL                               Description
Created                           Updated
-----
a52be735-30e5-4849-af23-83d65d592464 sample-broker-1 https://
demobroker.domain.com/ Service broker providing some valuable services
2018-06-22T13:04:19Z 2018-06-22T13:04:19Z
```

1.1.2.2.5.3.2 update-broker

Overview

smctl update-broker

Updates a service broker with the provided name in the SAP Service Manager instance.

Usage

```
smctl update-broker [name] <json_broker> [flags]
```

Example

```
smctl update-broker broker '{"name": "new-name", "description": "new-
description", "broker-url": "http://broker.com", "credentials": { "basic":
{ "username": "admin", "password": "admin" } } }'
```

Aliases

update-broker, ub

Parameters

Optional		Global Flag
-h, --help	Help for <code>update-broker</code> command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text.	No
--config	Set the path for the <code>smctl config.json</code> file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl update-broker sample-broker-1 '{"description": "Updated sample-broker
description"}'
ID                               Name
URL                               Description
Created                          Updated
-----
a52be735-30e5-4849-af23-83d65d592464 sample-broker-1 https://
demobroker.domain.com/ Updated sample-broker description 2018-06-22T13:04:19Z
2018-06-22T13:04:19Z
```

1.1.2.2.5.3.3 list-brokers

Overview

`smctl list-brokers`

Lists all service brokers registered in the SAP Service Manager instance.

Usage

```
smctl list-brokers [flags]
```

Aliases

list-brokers, lb

Parameters

Optional		Global Flag
-h, --help	Help for <code>list-brokers</code> command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text.	No
--config	Set the path for the <code>smctl config.json</code> file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl list-brokers
One broker registered.
ID                               Name
URL                               Description
Created                          Updated
-----
a52be735-30e5-4849-af23-83d65d592464 sample-broker-1 https://
demobroker.domain.com/ Service broker providing some valuable services
2018-06-22T13:04:19Z 2018-06-22T13:04:19Z
```

1.1.2.2.5.3.4 delete-broker

Overview

`smctl delete-broker`

Deletes a set of brokers registered in the SAP Service Manager instance.

Usage

```
smctl delete-broker [name] <name2 <name3> ... <nameN> [flags]
```

Aliases

delete-broker, db

Parameters

Optional		Global Flag
-h, --help	Help for <code>delete-broker</code> command.	No
--config	Set the path for the <code>smctl config.json</code> file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl delete-broker sample-broker-1  
Broker with name: sample-broker-1 successfully deleted
```

1.1.2.2.5.4 Instances

1.1.2.2.5.4.1 provision

Overview

`smctl provision`

Create a service instance in SAP Service Manager.

Usage

```
smctl provision [name] [offering] [plan] [flags]
```

Parameters

Optional		Global Flag
-h, --help	Help for <code>provision</code> command.	No
-b --broker-name	Name of the broker that provides the service offering. Only required when the offering name is ambiguous.	No
--mode	How calls to SAP Service Manager are performed. Possible values: sync or async (the default is async).	No
-c --parameters	A valid JSON object containing the instance parameters.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No
--config	Set the path for the smctl config.json file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

async execution:

```
> smctl provision sample-instance overview-service simple
Service Instance sample-instance successfully scheduled for provisioning. To see
status of the operation use:
smctl status /v1/service_instances/a6b0dfe6-1bd1-453f-a646-babd425b6b05/
operations/32bbbee7-a9d0-48e4-a434-bf47bc471a48
```

```
> smctl status /v1/service_instances/a6b0dfe6-1bd1-453f-a646-babd425b6b05/
operations/32bbbee7-a9d0-48e4-a434-bf47bc471a48
```

```
      | ID          | 32bbbee7-a9d0-48e4-a434-bf47bc471a48 |
      | Type         | create                                |
      | State        | succeeded                              |
```


sync execution:

```
> smctl provision sample-instance overview-service simple --mode sync
| ID | 0c170e73-28bd-47ea-b3f4-f1ad1dbf3e0a |
| Name | sample-instance |
| Service Plan ID | 25304783-2fc9-4f50-8dcb-0cbfe017ad15 |
| Platform ID | service-manager |
| Created | 2020-04-09T10:42:12.175051Z |
| Updated | 2020-04-09T10:42:13.2252101Z |
| Ready | true |
| Usable | true |
| Labels | subaccount_id=subacc-cfdev-tenant-id |
| Last Op | create succeeded |
```

1.1.2.2.5.4.2 get-instance

Overview

smctl get-instance

Get detailed information about a specified service instance.

Usage

```
smctl get-instance [name] [flags]
```

Aliases

get-instance, gi

Parameters

Optional		Global Flag
-h, --help	Help for <code>get-instance</code> command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No

Optional		Global Flag
<code>--show-instance-params</code>	Show the service instance configuration parameters.	No
<code>--config</code>	Set the path for the smctl config.json file (default is <code>\$HOME/.sm/config.json</code>).	Yes
<code>-v, --verbose</code>	Use verbose mode.	Yes

Example

```
> smctl get-instance sample-instance
One service instance.
| ID                | 0c170e73-28bd-47ea-b3f4-f1ad1dbf3e0a |
| Name              | sample-instance                       |
| Service Plan ID   | 25304783-2fc9-4f50-8dcb-0cbfe017ad15 |
| Platform ID       | service-manager                       |
| Created           | 2020-04-09T10:42:12.175051Z          |
| Updated           | 2020-04-09T10:42:13.22521Z          |
| Ready             | true                                   |
| Usable            | true                                   |
| Labels            | subaccount_id=subacc-cfdev-tenant-id |
| Last Op           | create succeeded                       |
```

```
> smctl get-instance sample-instance --show-instance-params
Showing parameters for service instance id: 0c170e73-28bd-47ea-b3f4-f1ad1dbf3e0a
The parameters are:
{
  "param1": "value1",
  "param2": "value2"
}
```

1.1.2.2.5.4.3 list-instances

Overview

smctl list-instances

Lists all service instances.

Usage

```
smctl list-instances [flags]
```

Aliases

list-instances, li

Parameters

Optional		Global Flag
-h, --help	Help for list-instances command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No
--config	Set the path for the smctl config.json file (default is \$HOME/.sm/config.json).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl list-instances
One service instance.
ID                               Name                               Service Plan
ID                               Created                            Labels
Updated                          Ready Usable
-----
-----
-----
0c170e73-28bd-47ea-b3f4-f1ad1dbf3e0a  sample-instance
25304783-2fc9-4f50-8dcb-0cbfe017ad15  service-manager
2020-04-09T10:42:12.175051Z  2020-04-09T10:42:13.22521Z  true  true
subaccount_id=subacc-cfdev-tenant-id
```

1.1.2.2.5.4.4 deprovision

Overview

smctl deprovision

Deletes a service instance.

Usage

```
smctl deprovision [name] [flags]
```

Parameters

Optional		Global Flag
-h, --help	Help for <code>deprovision</code> command.	No
-f, --force	Force delete - without confirmation.	No
-id	ID of the service instance. Required when the service instance name is ambiguous.	No
--mode	How calls to SAP Service Manager are performed. Possible values: sync or async (the default is async).	No
--config	Set the path for the smctl config.json file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

async execution:

```
> smctl deprovision sample-instance
Do you really want to delete instance with name [sample-instance] (Y/n): yes
Service Instance sample-instance successfully scheduled for deletion. To see
status of the operation use:
smctl status /v1/service_instances/0c170e73-28bd-47ea-b3f4-f1ad1dbf3e0a/
operations/40a748c1-c0f8-4acf-84a0-64e20914531d
```

Checking the status of the call:

```
> smctl status /v1/service_instances/0c170e73-28bd-47ea-b3f4-f1ad1dbf3e0a/
operations/40a748c1-c0f8-4acf-84a0-64e20914531d
| ID | 40a748c1-c0f8-4acf-84a0-64e20914531d |
| Type | delete |
| State | succeeded |
```

sync execution:

```
> smctl deprovision sample-instance --mode sync
Do you really want to delete instance with name [sample-instance] (Y/n): yes
Service Instance successfully deleted.
```

1.1.2.2.5.5 Bindings

1.1.2.2.5.5.1 bind

Overview

`smctl bind`

Creates a binding to a specific instance in the SAP Service Manager.

Usage

```
smctl bind [instance-name] [binding-name] [flags]
```

Parameters

Optional		Global Flag
-h, --help	Help for <code>bind</code> command.	No
--mode	How calls to SAP Service Manager are performed. Possible values: sync or async (the default is async).	No
-c, --parameters	A valid JSON object containing the instance parameters.	No
-id	ID of the service instance. Required when the service instance name is ambiguous.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No
--config	Set the path for the smctl config.json file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

async execution:

```
> smctl bind sample-instance sample-binding
Service Binding sample-binding successfully scheduled. To see status of the
operation use:
smctl status /v1/service_bindings/6372815d-29b8-4561-9898-016d6671b34b/
operations/ea67e94f-ad2f-4544-8b87-6924fe494327
```

Checking the status of the call:

```
> smctl status /v1/service_bindings/6372815d-29b8-4561-9898-016d6671b34b/
operations/ea67e94f-ad2f-4544-8b87-6924fe494327
| ID | ea67e94f-ad2f-4544-8b87-6924fe494327 |
| Type | create |
| State | succeeded |
```

sync execution:

```
smctl bind sample-instance sample-binding --mode sync
| ID | 5937785d-6740-4f56-bdd9-8d24544bddac |
| Name | sample-binding |
| Service Instance Name | sample-instance |
| Service Instance ID | 742b0c67-37f6-4c63-83d9-e3c5d2cb69f0 |
| Credentials | {"password": "pass", "username": "usr"} |
| Created | 2020-04-09T10:57:50.452161Z |
| Updated | 2020-04-09T10:57:51.5058215Z |
| Ready | true |
| Labels | subaccount_id=subacc-cfdev-tenant-id |
| Last Op | create succeeded |
```

1.1.2.2.5.5.2 get-binding

Overview

smctl get-binding

Get detailed information about a specific service binding.

Usage

```
smctl get-binding [name] [flags]
```

Aliases

get-binding, gsb

Parameters

Optional		Global Flag
-h, --help	Help for <code>get-binding</code> command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No
--show-binding-params	Show service binding configuration parameters.	No
--config	Set the path for the smctl config.json file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl get-binding sample-binding
One service binding.
| ID                | 5937785d-6740-4f56-bdd9-8d24544bddac |
| Name              | sample-binding                       |
| Instance Name     | sample-instance                      |
| Credentials       | {"password":"pass","username":"usr"} |
| Created           | 2020-04-09T10:57:50.452161Z          |
| Updated           | 2020-04-09T10:57:51.505822Z          |
| Ready             | true                                  |
| Labels            | subaccount_id=subacc-cfdev-tenant-id |
| Last Op           | create succeeded                      |
```

```
> smctl get-binding sample-binding --show-binding-params
Showing parameters for service binding id: 0c170e73-28bd-47ea-b3f4-f1ad1dbf3e0a
The parameters are:
{
  "param1":"value1",
  "param2":"value2"
}
```

1.1.2.2.5.5.3 list-bindings

Overview

smctl list-bindings

Lists all service bindings created in SAP Service Manager for the associated subaccount.

Usage

```
smctl list-bindings [flags]
```

Aliases

list-bindings, lsb

Parameters

Optional		Global Flag
-h, --help	Help for list-bindings command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No
--config	Set the path for the smctl config.json file (default is \$HOME/.sm/config.json).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl list-bindings
One service binding.
ID                               Name                               Instance Name
Credentials                      Ready Labels                        Created
Updated
```



```

-----
-----
5937785d-6740-4f56-bdd9-8d24544bddac sample-binding sample-instance
{"password":"pass","username":"usr"} 2020-04-09T10:57:50.452161Z
2020-04-09T10:57:51.505822Z true subaccount_id=subacc-cfdev-tenant-id

```

1.1.2.2.5.5.4 unbind

Overview

smctl unbind

Deletes a service binding with the name provided for a specified service instance.

Usage

```
smctl unbind [instance-name] [binding-name] [flags]
```

Parameters

Optional		Global Flag
-h, --help	Help for <code>unbind</code> command.	No
-f, --force	Force delete - without confirmation.	No
-id	ID of the service binding. Required when the service binding name is ambiguous.	No
--mode	How calls to SAP Service Manager are performed. Possible values: sync or async (the default is async).	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No
--config	Set the path for the smctl config.json file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

async execution:

```
> smctl unbind sample-instance sample-binding
Do you really want to delete binding with name [sample-binding] for instance
with name sample-instance (Y/n): yes
Service Binding sample-binding successfully scheduled for deletion. To see
status of the operation use:
smctl status /v1/service_bindings/5937785d-6740-4f56-bdd9-8d24544bddac/
operations/6066bd46-79d4-4f8e-be50-9ad2e5ca035a
```

Checking the status of the call:

```
> mctl status /v1/service_bindings/5937785d-6740-4f56-bdd9-8d24544bddac/
operations/6066bd46-79d4-4f8e-be50-9ad2e5ca035a
| ID          | 6066bd46-79d4-4f8e-be50-9ad2e5ca035a |
| Type        | delete                                |
| State       | succeeded                              |
```

sync execution:

```
> smctl unbind sample-instance sample-binding --mode sync
Do you really want to delete binding with name [sample-binding] for instance
with name sample-instance (Y/n): yes
Service Binding successfully deleted.
```

1.1.2.2.5.6 Offerings

1.1.2.2.5.6.1 list-offerings

Overview

smctl list-offerings

Lists all service offerings that are associated with the SAP Service Manager for this subaccount.

Usage

```
smctl list-offerings [flags]
```

Aliases

list-offerings, lo

Parameters

Optional		Global Flag
-h, --help	Help for <code>list-offerings</code> command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No
--config	Set the path for the <code>smctl config.json</code> file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl list-offerings
One service offering.
ID                                     Name
Description                           Ready  Labels
-----
54944d91-75b9-442c-aecd-f98821490740 overview-service Provides an overview of
46343c8e-957f-4fde-8176-ca3510d489e0 true any service instances and bindings that have been created by a platform.
```

1.1.2.2.5.6.2 list-plans

Overview

`smctl list-plans`

Lists all service plans that are associated with the SAP Service Manager for this subaccount.

Usage

```
smctl list-plans [flags]
```

Parameters

Optional		Global Flag
-h, --help	Help for <code>list-plans</code> command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No
--config	Set the path for the <code>smctl config.json</code> file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes
-e, --environment	Shows plans by specified environments. Valid values: Cloud Foundry, Kubernetes.	No
-f, --field--query	Filters plans by field query.	
-l, --label-query	Filters fields by label query.	

Example

```
> smctl list-plans
2 service plans.
ID                               Name      Description
Offering ID                      Ready    Labels
-----
aec1cdac-9faa-4aa4-aeb7-6dbcc275208d simple    A very simple plan.
a56dc9b4-70f9-45e3-a8a1-3b3a06289aa5 true
eb2ce3e0-d64c-4977-b51e-2f682ec2d835 complex  A more complicated plan.
a56dc9b4-70f9-45e3-a8a1-3b3a06289aa5 true
```

1.1.2.2.5.6.3 marketplace

Overview

`smctl marketplace`

Lists all service offerings with their service plans that are available in SAP Service Manager for this subaccount.

Usage

```
smctl marketplace [flags]
```

Aliases

marketplace, m

Parameters

Optional		Global Flag
-h, --help	Help for <code>marketplace</code> command.	No
-s, --service	Detailed information about the plans of a specific service offering.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No
--config	Set the path for the <code>smctl config.json</code> file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl marketplace
One service offering.
```

Name	Plans
Description	Broker ID
-----	-----
-----	-----
overview-service	simple, complex
Provides an overview of any service instances and bindings that have been created by a platform.	46343c8e-957f-4fde-8176-ca3510d489e0

```
> smctl marketplace -s overview-service
2 service plans for this service offering.
Plan      Description          ID
-----
simple     A very simple plan.  25304783-2fc9-4f50-8dcb-0cbfe017ad15
complex   A more complicated plan. 52207e8e-1456-4f2e-b3df-3b97fe8d3d6f
```

1.1.2.2.5.7 Other Commands

1.1.2.2.5.7.1 status

Overview

smctl status

Get the status of an asynchronous operation.

Usage

```
smctl status operation URL path [flags]
```

Parameters

Optional		Global Flag
-h, --help	Help for status command.	No
-o, --output	Output format of the command. Possible options: json, yaml, text	No

Optional		Global Flag
--config	Set the path for the smctl config.json file (default is \$HOME/.sm/config.json).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl status /v1/service_bindings/5937785d-6740-4f56-bdd9-8d24544bddac/
operations/6066bd46-79d4-4f8e-be50-9ad2e5ca035a
| ID      | 6066bd46-79d4-4f8e-be50-9ad2e5ca035a |
| Type    | delete                               |
| State   | succeeded                             |
```

1.1.2.2.5.7.2 info

Overview

smctl info

Displays information about the SAP Service Manager instance to which the **SMCTL** is connected.

Usage

```
smctl info [flags]
```

Aliases

```
info, i
```

Parameters

Optional		Global Flag
-h, --help	Help for info command.	No
--config	Set the path for the smctl config.json file (default is \$HOME/.sm/config.json).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl info
Service Management URL: https://service-management-url.com/
Logged user: testUser
```

1.1.2.2.5.7.3 version

Overview

smctl version

Displays information about the smctl version in use.

Usage

```
smctl version [flags]
```

Aliases

```
version, v
```


Parameters

Optional		Global flag
-h, --help	Help for <code>version</code> command.	No
--config	Set the path for the <code>smctl config.json</code> file (default is <code>\$HOME/.sm/config.json</code>).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl version
Service Management Client 0.0.1
```

1.1.2.2.5.7.4 help

Overview

`smctl help`

Displays the whole list of available commands. When a specific command is provided, help information for the specified command is displayed.

Usage

```
smctl help [command] [flags]
```

Aliases

help, -h

Parameters

Optional		Global Flag
--config	Set the path for the smctl config.json file (default is \$HOME/.sm/config.json).	Yes
-v, --verbose	Use verbose mode.	Yes

Example

```
> smctl help
smctl controls a Service Management instance.
Usage:
  smctl [command]
Available Commands:
  delete-broker      Deletes brokers
  delete-platform    Deletes platforms
  help               Help about any command
  info               Prints information for logged user
  list-brokers       List brokers
  list-platforms     List platforms
  login              Logs user in
  register-broker    Registers a broker
  register-platform  Registers a platform
  update-broker      Updates broker
  update-platform    Updates platform
  version            Prints smctl version
Flags:
  --config string    config file (default is $HOME/.sm/config.json)
  -h, --help         help for smctl
  -v, --verbose      verbose
Use "smctl [command] --help" for more information about a command.
```

1.1.2.3 Working with SAP Service Manager APIs

The SAP Service Manager service API defines a centralized REST interface that allows the management of platforms, service brokers, service offerings, service plans, service instances, and service bindings.

API Specifications in SAP Business Accelerator Hub

Visit [SAP Business Accelerator Hub](#) to view the list of all SAP Service Manager APIs, their specifications, and links to accompanying documentation.

You can also try out the APIs.

Related Information

[Accessing the APIs \[page 59\]](#)

[Rate Limiting \[page 67\]](#)

1.1.2.3.1 Accessing the APIs

You can access the SAP Service Manager service APIs with a valid OAuth2 access token. The API access level (read/write) and the allowed API endpoints are controlled by the access token scopes.

There are two methods of obtaining access tokens:

- User Access Tokens – represents a named user. The included scopes are derived from the user's roles. See [User Access \[page 59\]](#).
- Client Access Tokens – represents a technical client. Such tokens are retrieved using OAuth client credentials provided by instances of the service-manager service. The included scopes are determined by the used service plan. See [Technical Access \[page 61\]](#).

1.1.2.3.1.1 User Access

The SAP Service Manager API can be accessed using OAuth 2.0 access tokens issued for named users, which have the proper roles assigned to them.

Prerequisites

- Obtain the credentials for SAP BTP cockpit to access services from SAP Business Technology Platform in your subaccount. See [Subscribing to SAP Service Manager \[Feature Set A\] \[page 21\]](#).

i Note

If you are using cloud management tools feature set B, you don't to obtain the credentials to subscribe to SAP Service Manager because the required role collections are automatically available to you without the whole subscription process.

- Assign the SAP Service Manager roles to the user. See [Assign the Subaccount Service Administrator Collection \[page 22\]](#).
For more information about the scopes included in each role, see [SAP Service Manager Roles \[page 60\]](#).

Procedure

1. Get an access token by running the following command:

Linux:

```
curl 'https://service-manager.cfapps.<region domain>/v1/oauth/<subdomain>/token' \
-i -X POST \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'Accept: application/json' \
--data 'grant_type=password&username=<username>&password=<password>'
```

Windows::

```
curl ^
-i -X POST ^
-H "Content-Type: application/x-www-form-urlencoded" ^
-H "Accept: application/json" ^
-H "Authorization: Basic Y2Y6" ^
--data "grant_type=password&username=<username>&password=<password>" https://service-manager.cfapps.sap.hana.ondemand.com/v1/oauth/<subdomain>/token
```

i Note

If in Swagger, use your username and password. The subdomain is the subaccount subdomain, in which you would like to try the API. You can find it in the SAP BTP cockpit [Overview](#) tab of your subaccount.

See [Enable API Access to an XSUAA Configuration](#).

i Note

The access token received also contains the scopes that are granted for this access token. Therefore, only APIs which require one of these scopes can be used with this access token.

```
{
  "access_token": "<access_token>",
  "token_type": "bearer",
  "expires_in": 43199,
  "scope": "<xsappname>.job.read <xsappname>.event.read"
}
```

2. Add the Authorization header to your request: "Authorization: Bearer <access token>".

1.1.2.3.1.1.1 SAP Service Manager Roles

Describes the SAP Service Manager roles.

Roles	Description	Scopes
Subaccount Service Administrator	Allows to manage resources in the subaccount in which the <i>service-manager</i> instance of this plan was created. This includes managing subaccount scoped brokers, platforms, instances and bindings and also reading services, plans and visibilities.	<ul style="list-style-type: none"> • subaccount.broker.manage • subaccount.broker.read • subaccount.platform.manage • subaccount.platform.read • subaccount.service_instance.read • subaccount.service_instance.manage • subaccount.service_binding.read • subaccount.service_binding.manage • subaccount.service_plan.read • subaccount.service_offering.read
Subaccount Service Viewer [Feature Set B]	Allows read-only access to the resources in the subaccount in which the <i>service-manager</i> instance was created. This includes reading subaccount scoped brokers, platforms, instances and bindings, services, plans, and visibilities.	<ul style="list-style-type: none"> • subaccount.broker.read • subaccount.platform.read • subaccount.service_instance.read • subaccount.service_binding.read • subaccount.service_plan.read • subaccount.service_offering.read

1.1.2.3.1.2 Technical Access

The SAP Service Manager API can be accessed using OAuth 2.0 access tokens issued for technical clients by providing the client credentials (ID and secret).

Client credentials can be obtained by creating a `service-manager` service instance and binding.

- [Create a SAP Service Manager Instance and Binding \[page 61\]](#)
- [Retrieve an OAuth 2.0 Access Token \[page 66\]](#)

1.1.2.3.1.2.1 Create a SAP Service Manager Instance and Binding

The SAP Service Manager APIs are protected with OAuth 2.0 client credentials. This document describes the steps you need to perform to create an OAuth client and obtain an access token to call the SAP Service Manager APIs, and shows a detailed example of the procedure in the Cloud Foundry environment.

Procedure

1. Create a SAP Service Manager service instance with one of the [SAP Service Manager Broker Plans \[page 65\]](#).

For more information about how to create an instance, see [Creating Service Instances \[page 9\]](#).

2. Create a binding object (service key in the Cloud Foundry environment) for the SAP Service Manager instance.

There are two types of binding objects, depending on the type of authorization you choose:

- Default credentials type binding
- X.509 credentials type binding

To create a binding object with the default credentials type, run the following command:

```
smctl bind <sm_instance_name> <sm_binding_name>
```

- `<sm_instance_name>` - the name of the service instance for which you're creating the binding.
- `<sm_binding_name>` - the name of the service binding you're creating.

The example of a binding object created with the default credentials type:

```
{
  "clientid": "<client_id>",
  "clientsecret": "<client_secret>",
  "sm_url": "<service_manager_URL>",
  "url": "<https://<subdomain>.authentication.<region domain>",
  "xsappname": "<xsapp name>"
}
```

To create a binding object with the X.509 credentials type, run the following command by using the Service Manager Control (SMCTL) command-line tool. For more information about SMCTL, see [Using the Service Manager Control \(SMCTL\) Command-Line Tool \[page 20\]](#):

```
smctl bind <sm_instance_name> <sm_binding_name> -c {"credential-type":"x509"}
```

Note

The command example shows minimum requirements for JSON object parameters specification. Besides the `credential-type`, you can also provide the following parameters in the same object:

- `key-length` - specifies the byte length of the generated private key. Default value is 2048 bytes.
- `validity-type` - specifies the validity time unit. Valid values are: `DAYS`, `MONTHS`, and `YEARS`. Default value is `DAYS`.
- `validity` - specifies the number of time units in `validity-type`. Default value is 7, thus the complete validity defaults to 7 `DAYS`.

The example of the JSON object with all of the parameters specified:

```
{
  "credential-type": "x509",
  "x509": {
    "key-length": 2048,
    "validity": 7,
    "validity-type": "DAYS"
  }
}
```

The example of a binding object created with the X.509 credentials type:

```
{
  "clientid": "xxxxxxx",
  "certificate": "-----BEGIN CERTIFICATE-----...-----END CERTIFICATE",
  "key": "-----BEGIN RSA PRIVATE KEY-----...-----END RSA PRIVATE KEY-----",
}
```

```
"certurl": "<certificate_URL>",
"xsappname": "<name>",
"sm_url": "<service_manager_URL>"
}
```

For more information about the required steps for different environment types, see [Consuming SAP BTP Services from Various Environments \[page 73\]](#).

Obtaining Access Credentials in Cloud Foundry

Prerequisites

- You have downloaded and installed the latest version of the Cloud Foundry command line interface (cf CLI). See [Download and Install the Cloud Foundry Command Line Interface](#).
- You have a Cloud Foundry org which has service access to at least one **service-manager** service plan.

Note

To check if you have an access to Service Manager (technical name: **service-manager**) in the Cloud Foundry marketplace, execute the following command:

```
cf marketplace -s service-manager
```

You should see the **service-manager** service plans to which your orgs have access.

- You have created a space in your Cloud Foundry org.

Procedure

1. Login to your space using cf CLI. See [Log On to the Cloud Foundry Environment Using the Cloud Foundry Command Line Interface](#).

Example:

```
cf api https://api.<landscape domain>
cf login
cf target -o ORG -s SPACE
```

2. Using the cf CLI run the following command to create a service instance. See [Create Service Instances Using the Cloud Foundry Command Line Interface](#)

```
cf create-service service-manager PLAN SERVICE_INSTANCE
```

Specify the following parameters:

- **PLAN**: The name of the service plan you want to use. See [SAP Service Manager Broker Plans \[page 65\]](#).
 - **SERVICE_INSTANCE**: Name of the service instance.
3. Get the OAuth 2.0 client information and SAP Service Manager endpoints using either one of the following methods:
 1. Bind the service instance to your application. See [Bind Service Instances to Applications Using the Cloud Foundry Command Line Interface](#).
 - Display the environment of your application.

```
cf env APP-NAME
```

- Locate the details of your SAP Service Manager instance in the application environment, for example:

```
System-Provided:
{
  "VCAP_SERVICES": {
    ...
    "service-manager": [
      {
        "credentials": {
          "clientid": "<client_id>",
          "clientsecret": "<client_secret>",
          "sm_url": "<service management URL>",
          "url": "<token URL>",
          "xsappname": "<xsapp name>"
        },
        "instance_name": "<SERVICE_INSTANCE>",
        "plan": "<PLAN>"
      }
    ]
    ...
  }
}
```

2. Create a service key. See [Create Service Keys Using the Cloud Foundry Command Line Interface](#).

```
cf create-service-key SERVICE_INSTANCE SERVICE_KEY
```

Specify the following parameters:

- **SERVICE_INSTANCE**: Name of the service instance.
- **SERVICE_KEY**: Name for the service key.

Display the service key information:

```
cf service-key SERVICE_INSTANCE SERVICE_KEY
{
  "clientid": "<client_id>",
  "clientsecret": "<client_secret>",
  "sm_url": "<service management URL>",
  "url": "<https://<subdomain>.authentication.<region domain>>",
  "xsappname": "<xsapp name>"
},
```

Working With Swagger APIs

Use the values of the parameters: `clientid`, `clientsecret`, and the `<subdomain>` part of the `url` parameter's value you obtained in the previous step to authenticate with the SAP Service Manager.

For example, in the following output:

```
{
  "clientid": "sb-fd7c7fac-61d5-4871-bce6-f82584abea4e!b4065|service-manager!b4065",
  "clientsecret": "*****Qa7tJPIu*****",
  "sm_url": "https://service-manager.cfapps.sap.hana.ondemand.com",
  "url": "https://svcmgr.authentication.sap.hana.ondemand.com",
  "xsappname": "fd7c7fac-61d5-4871-bce6-f82584abea4e!b4065|service-manager!b4065"
}
```

the needed values are:

- sb-fd7c7fac-61d5-4871-bce6-f82584abea4e!b4065|service-manager!b4065 for clientid
- *****Qa7tJPIu***** for clientsecret (only part of the value is exposed in the example)
- svcmgr in the url

Go to <https://service-manager.<app domain>.<landscape domain>/swaggerui/swagger-ui.html>

❁ Example

If your account is running on the Europe (Frankfurt) region, use this URL:

<https://service-manager.cfapps.eu10.hana.ondemand.com/swaggerui/swagger-ui.html>

1.1.2.3.1.2.1.1 SAP Service Manager Broker Plans

Describes the SAP Service Manager plans.

Broker Plans	Description	Scopes
subaccount-admin	Allows to manage resources in the subaccount in which the <i>service-manager</i> instance of this plan was created. This includes managing subaccount scoped brokers, platforms, instances and bindings and also reading services, plans and visibilities.	<ul style="list-style-type: none"> • subaccount.broker.manage • subaccount.broker.read • subaccount.platform.manage • subaccount.platform.read • subaccount.service_instance.read • subaccount.service_instance.manage • subaccount.service_binding.read • subaccount.service_binding.manage • subaccount.service_plan.read • subaccount.service_offering.read
subaccount-audit	Allows read-only access to the resources in the subaccount in which the <i>service-manager</i> instance was created. This includes reading subaccount scoped brokers, platforms, instances and bindings, services, plans, and visibilities.	<ul style="list-style-type: none"> • subaccount.broker.read • subaccount.platform.read • subaccount.service_instance.read • subaccount.service_binding.read • subaccount.service_plan.read • subaccount.service_offering.read

Broker Plans	Description	Scopes
container	Allows management of service instances and bindings in a reduced scope that corresponds to the service instance. Instances created using the binding credentials of the container instance are visible from the instance itself and from instances of the subaccount-* plans, but not from other container instances.	<ul style="list-style-type: none"> • subaccount.service_instance.read • subaccount.service_instance.manage • subaccount.service_binding.read • subaccount.service_binding.manage • subaccount.service_plan.read • subaccount.service_offering.read
service-operator-access	Provides credentials for SAP BTP service operator to access SAP BTP from a Kubernetes cluster.	

1.1.2.3.1.2.2 Retrieve an OAuth 2.0 Access Token

1. Get an access token:

Use the `url`, `clientid`, and `clientsecret` you obtained when you created a service instance ([Create a SAP Service Manager Instance and Binding \[page 61\]](#)) with one of the broker plans ([SAP Service Manager Broker Plans \[page 65\]](#)) to request an access token using the following command:

```
curl '<url>/oauth/token' -X POST \
  -H 'Accept: application/json' \
  -d
'grant_type=client_credentials&client_id=<clientid>&client_secret=<clientsecret>'
```

iNote

The access token received also contains the scopes that are granted for this access token. Therefore, only APIs which require one of these scopes can be used with this access token.

```
{
  "access_token": "<access_token>",
  "token_type": "bearer",
  "expires_in": 43199,
  "scope": "<xsappname>.job.read <xsappname>.event.read"
}
```

See [Enable API Access to an XSUAA Configuration](#).

2. Add the Authorization header to your request: "Authorization: Bearer <access token>"

1.1.2.3.2 Rate Limiting

Describes how all API requests to the SAP Service Manager adhere to rate-limiting rules.

About API Rate Limiting

API rate limiting is, in a nutshell, limiting people (and bots) from accessing the API based on the rules set by the API's operator or owner.

API rate limiting can be used as a defensive security measure for the API, and also a quality control method. As a shared service, the API must protect itself from excessive use to encourage an optimal experience for anyone using the API. Quality-wise, as all APIs operate on finite resources, rate limiting is essential to improve the availability of API service for many users as possible by avoiding excessive resource usages.

All APIs define their own custom rate-limit rules for the number of requests per time window and per identified caller.

API callers are identified through the authenticated requests associated with the username on the authenticated platform or with the OAuth client ID.

When the rate limit is exceeded, the client receives the [HTTP 429 Too Many Requests](#) response status code.

Service Manager APIs

The Service Manager works with the following resources: platforms, service brokers, service bindings, service offerings, service plans, and service instances.

There's a dedicated group of APIs for each of the SAP Service Manager resources.

The total number of requests that you can perform for all SAP Service Manager APIs together is 10,000 calls per hour and 1000 per minute.

While the same rate-limiting rule per hour and minute applies to all platform and service broker APIs, as well as all available GET APIs for the service instances group, different, and more restrictive rules apply to other Service Manager resource API groups. Refer to the details in the table below.

→ Remember

All API calls executed in the same time frame (minute and hour) **count together** towards rate limits. See the examples below the table for more details.

API Endpoint	Maximum Number of Calls per Timeframe
/v1/service_bindings	<ul style="list-style-type: none">Hour: 6000Minute: 600

API Endpoint	Maximum Number of Calls per Timeframe
/v1/service_offerings	<ul style="list-style-type: none"> Hour: 1000 Minute: 100
/v1/service_plans	<ul style="list-style-type: none"> Hour: 1000 Minute: 100
/v1/service_instances	<ul style="list-style-type: none"> Hour: 6000 Minute: 600

Note

CREATE /v1/service_instances is limited to 50 calls per minute.

❖ Example

- You've called one of the /v1/service_offerings APIs 100 times within a minute. If you try to call any of the APIs in that same resource group again, you get the HTTP 429 response code. However, you can still call any of the /v1/service_bindings APIs up to 500 times within that same minute because their per-minute limit is 600 and you've already used 100 of them on another API resource group in that minute.
- Let's say you've now used up all of the remaining 500 API calls for /v1/service_bindings APIs for that minute. If you try to call any of the APIs in that same resource group again within the same minute, you get the HTTP 429 response code. However, you still have up to 400 available API calls for any of the resource groups to which the 1000-per-minute rule applies because $1000 - 600$ (used calls) = 400.
- If you used all available 1000 API calls in that minute, for the same hour you still have up to: 10,000 available API calls for the hour - 1000 used API calls = 9000 API calls.

Response

The error you receive after calling one of the /v1/service_offerings APIs 100 times within a minute and then one of the /v1/service_bindings 501 times within that same minute:

```
HTTP/1.1 429 Too Many Requests
{
  "description": "The allowed request limit of 600 responses has been reached.
Please try again later.Check the 'Retry-After' header value to see how long you
need to wait."
}
```

The example shows that there's also the `Retry-After` header value at your disposal. It indicates how long you need to wait before you can try again.

The `Retry-After` header value is in HTTP-date format:

Date: <day-name>, <day> <month> <year> <hour>:<minute>:<second> GMT

`Retry-After`

1.1.2.3.3 Filtering Parameters and Operators

Use various combinations of filtering parameters to optimize the results of the Get all API calls of the SAP Service Manager service resources APIs.

The resources to which these parameters apply are: platforms, service brokers, service instances, service bindings, service plans, and service offerings.

Check the SAP Service Manager API section of [SAP Business Accelerator Hub](#) for more details about the Get all APIs for each of the resources.

Context

There are two types of filtering based on resource fields and labels.

You can control filtering with the following query parameters:

Parameter Name	Parameter Type	Description
fieldQuery	string	<p>Returns the items that have field values that match the provided field query.</p> <p>Must be a nonempty string.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>❖ Example</p> <pre>type eq 'kubernetes'</pre> </div>
labelQuery	string	<p>Returns the items that have label values that match the provided label query.</p> <p>Must be a nonempty string.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>❖ Example</p> <pre>environment eq 'dev'</pre> </div>

Field and label values that are used to filter the SAP Service Manager resources in API calls are called literals.

The SAP Service Manager APIs support the following types of literals:

- string
- boolean
- integer
- date-time

Filter Syntax and Rules

- String literals must be enclosed in single quotes (' '). Single quotes in strings must be encoded with another single quote (' ' ').
For example:
`type ne 'cloud foundry'`
`description eq ''customized' kubernetes'`
- Boolean literals can't be enclosed in quotes.
- Literals can't be enclosed in brackets.
- Integer literals can only consist of digits with one optional leading + or - sign.
- Date-time literals must follow ISO 8601 format, and can't be enclosed in quotes.
The supported ISO 8601 format is `yyyy-mm-ddThh:mm:ss.s[Z|(+|-)hh:mm]`

Supported Query Operators

The table shows all the supported operators that you can use to filter the results of your API calls.

Operator	Field Query	Label Query
<code>eq</code> (equal)	Evaluates to true if the field value matches the literal. False otherwise.	Evaluates to true if the label exists and one label value matches the literal. False otherwise.
<code>en</code> (equal or null)	Evaluates to true if the field value matches the literal or if the field value is <code>null</code> . False otherwise.	Evaluates to true if the label exists and one label value matches the literal, or if the label doesn't exist. False otherwise.
<code>ne</code> (not equal)	Evaluates to true if the field value doesn't match the literal. False otherwise.	Evaluates to true if the label exists and no label value matches the literal. False otherwise.
<code>in</code>	Evaluates to true if the field value matches at least one value in the list of literals. False otherwise.	Evaluates to true if the label exists and the label value matches at least one value in the list of literals. False otherwise.
<code>notin</code>	Evaluates to true if the field value doesn't match any value in the list of literals. False otherwise.	Evaluates to true if the label exists and no label value matches any value in the list of literals. False otherwise.
<code>and</code>	Evaluates to true if both the left and right operands evaluate to true. False otherwise.	Evaluates to true if both the left and right operands evaluate to true. False otherwise.
<code>contains</code>	Evaluates to true if the literal is part of the field value. False otherwise.	Evaluates to true if the literal is part of the label value. False otherwise.

The following operators apply only to the field query:

Operator	Field Query
gt	Evaluates to true if the field value is greater than the literal. False otherwise.
ge	Evaluates to true if the field value is greater than, or equal to the literal. False otherwise.
lt	Evaluates to true if the field value is less than the literal. False otherwise.
le	Evaluates to true if the field value is less than, or equal to the literal. False otherwise.

Note

Label and field queries can be combined. The returned lists only contain entries that match both queries.

Query Task Examples

Field Query

- List all service instances with a service plan ID that equals `bvsded31-c303-123a-aab9-8crar19e1218`.

```
service_plan_id eq 'bvsded31-c303-123a-aab9-8crar19e1218'
```

- List all bindings that use the `small` or `medium` plan of the `mysql` service provided by the broker with the ID `f85bcbd3-6c8b-43f0-a019-7f0a1ec5dba4`.

```
broker_id eq 'f85bcbd3-6c8b-43f0-a019-7f0a1ec5dba4' and service_name eq 'mysql' and plan_name in ('small', 'medium')
```

- List all service instances that aren't of the `postgresql` service, that are managed by the SAP Service Manager, and that aren't orphans.

```
platform_id eq 'service-manager' and service_name ne 'postgresql' and orphan ne true
```

Label Query

- List all service instances with a `context_id` label that has a value `ad8cddb0-4679-43bf-89bc-357e9a638f30`.

```
context_id eq 'ad8cddb0-4679-43bf-89bc-357e9a638f30'
```

Combined Field and Label Queries

- List all kubernetes platforms whose purpose is `dev`.

i Note

The following code samples are field and label queries respectively.

```
type eq 'kubernetes'
```

```
purpose eq 'dev'
```

1.1.2.4 SAP Service Manager Commands for SAP BTP Command Line Interface (btp CLI) [Feature Set B]

Use the SAP BTP command line interface to work with the SAP Service Manager resources.

i Note

The content in this section is only relevant for cloud management tools feature set B.

Learn how to use the client:

- [Download and Start Using the Client](#)
- [Set the Default Command Context](#)

→ Tip

All of these commands are executed for a specific subaccount. We recommend using the `btp target` command to set the default context to a subaccount. See [Set a Target for Subsequent Commands with btp target](#).

SAP Service Manager btp CLI commands

For the full list of the available commands for SAP Service Manager and details about them, see the **services/** sections of the [btp CLI Command Reference](#) topic.

Related Information

[Account Administration Using the SAP BTP Command Line Interface \(btp CLI\)](#)

1.2 Consuming SAP BTP Services from Various Environments

Learn more about how to consume SAP BTP services from various runtime environments.

[Consuming SAP BTP Services from the Cloud Foundry Environment \[page 73\]](#)

SAP BTP, Cloud Foundry environment is an open Platform-as-a-Service (PaaS) targeted at microservice development and orchestration.

[Consuming SAP BTP Services from the Neo Environment \[page 74\]](#)

SAP BTP, Neo environment is an enterprise platform-as-a-service (enterprise PaaS) that provides comprehensive application development services and capabilities, which lets you build, extend, and integrate business applications in the cloud.

[Consuming SAP BTP Services from the Kyma Environment \[page 74\]](#)

SAP BTP Kyma environment is a fully managed Kubernetes-based environment that allows you to consume external services and use their functionality to build and deploy your own applications.

[Consuming Services in Other Environments Using the SAP Service Manager Instances \[page 75\]](#)

Consume SAP BTP services from any runtime environment by creating service instances and service bindings directly in your subaccount with the Service Manager Control (SMCTL) CLI or APIs.

[Consuming SAP BTP Services from Kubernetes \[page 78\]](#)

Kubernetes, also known as K8s, is an open-source environment for automating deployment, scaling, and management of containerized applications.

Parent topic: [Consuming Services in SAP BTP \[page 3\]](#)

Related Information

[SAP Service Manager \[page 3\]](#)

1.2.1 Consuming SAP BTP Services from the Cloud Foundry Environment

SAP BTP, Cloud Foundry environment is an open Platform-as-a-Service (PaaS) targeted at microservice development and orchestration.

To learn more about using services in the SAP BTP, Cloud Foundry environment, such as how to create (user-provided) service instances, bind them to applications, and how to create service keys, see [Using Services in the Cloud Foundry Environment](#).

Parent topic: [Consuming SAP BTP Services from Various Environments \[page 73\]](#)

Related Information

[Consuming SAP BTP Services from the Neo Environment \[page 74\]](#)

[Consuming SAP BTP Services from the Kyma Environment \[page 74\]](#)

[Consuming Services in Other Environments Using the SAP Service Manager Instances \[page 75\]](#)

[Consuming SAP BTP Services from Kubernetes \[page 78\]](#)

1.2.2 Consuming SAP BTP Services from the Neo Environment

SAP BTP, Neo environment is an enterprise platform-as-a-service (enterprise PaaS) that provides comprehensive application development services and capabilities, which lets you build, extend, and integrate business applications in the cloud.

SAP BTP services in the Neo environment don't implement the Open Service Broker API, and therefore aren't managed by SAP Service Manager service.

For more information, see [Using Services in the Neo Environment](#).

Parent topic: [Consuming SAP BTP Services from Various Environments \[page 73\]](#)

Related Information

[Consuming SAP BTP Services from the Cloud Foundry Environment \[page 73\]](#)

[Consuming SAP BTP Services from the Kyma Environment \[page 74\]](#)

[Consuming Services in Other Environments Using the SAP Service Manager Instances \[page 75\]](#)

[Consuming SAP BTP Services from Kubernetes \[page 78\]](#)

1.2.3 Consuming SAP BTP Services from the Kyma Environment

SAP BTP Kyma environment is a fully managed Kubernetes-based environment that allows you to consume external services and use their functionality to build and deploy your own applications.

Read more about creating service instances and creating connections between instances and applications: [Using Services in the Kyma Environment](#)

For an example of using the SAP S/4HANA Cloud Extensibility Service, see [Extending SAP S/4HANA Cloud in the Kyma Environment](#).

Parent topic: [Consuming SAP BTP Services from Various Environments \[page 73\]](#)

Related Information

[Consuming SAP BTP Services from the Cloud Foundry Environment \[page 73\]](#)

[Consuming SAP BTP Services from the Neo Environment \[page 74\]](#)

[Consuming Services in Other Environments Using the SAP Service Manager Instances \[page 75\]](#)

[Consuming SAP BTP Services from Kubernetes \[page 78\]](#)

1.2.4 Consuming Services in Other Environments Using the SAP Service Manager Instances

Consume SAP BTP services from any runtime environment by creating service instances and service bindings directly in your subaccount with the Service Manager Control (SMCTL) CLI or APIs.

Context

The services are consumed in the context of your SAP BTP subaccount, which is necessary for verifying the required entitlements to commercial services and all the related administrative operations such as billing of the services that incur costs.

Consumption Flow

i Note

The following example demonstrates how to consume an SAP BTP service by using either the Service Manager Control (SMCTL) CLI or APIs via curl.

To learn how to consume services from other environments by using the SAP BTP cockpit, see the following topics:

- [Creating Instances in Other Environments \[page 13\]](#)
- [Creating Service Bindings in Other Environments \[page 18\]](#)

By using CLI or API, you achieve the integration with any deployment script that supports the execution of shell scripts or commands.

For more information about how to use the SAP Service Manager service CLI and API, see the following links: [Using the Service Manager Control \(SMCTL\) Command-Line Tool \[page 20\]](#) and [Working with SAP Service Manager APIs \[page 58\]](#).

Example

In the example, you use the `application` plan of the `xsuaa` service. The specific plan and service are convenient for demonstration since they don't require any entitlements and don't incur costs.

i Note

You can apply the same steps to any other service that supports cross consumption and that your subaccount is entitled to use.

General Prerequisites

You have an SAP Business Technology Platform subaccount.

Consuming Services with the Service Manager Control (SMCTL) CLI

Prerequisites

- You're logged on to your subaccount.
For more information, see [Using the Service Manager Control \(SMCTL\) Command-Line Tool \[page 20\]](#).
 - You have a command-line JSON processor at your disposal, for example jq.
1. Locate the plan for the service you want to use by running the following command that lists all the available services in your subaccount:

```
smctl marketplace
```

The output includes the name of the service, the plan used for the service, the description of the service, and the ID of the service broker associated with it.

In the example, we choose the `xsuaa` service with the `application` plan.

2. Create a service instance of the selected `xsuaa` service by running the following command:

```
smctl provision example-xsuaa-instance xsuaa application --param async=false
```

Where the name of the new instance is `example-xsuaa-instance`, and the `--param async=false` setting indicates that the command output becomes available only once the instance is ready to be used.

3. Create a binding to access the service instance you created in the previous step by running the following command:

```
smctl bind example-xsuaa-instance example-xsuaa-binding --param async=false
```

The name of the new binding is `example-xsuaa-binding`, and the output includes the `credentials` section that contains the access URLs and authorization details required to use the `xsuaa` service.

4. Inject service access details to your application.

i Note

While the method to extract the relevant access and authorization details from the binding credentials is universal for all environments, the way to inject those details to an application is specific to the environment in which the application is deployed.

For example, if you're deploying a Spring Boot Java application, you can use the `application.properties` file or the Java environment to inject the required values.

In this example, we extract the `clientId` field from the binding credentials, and store it in the `client_id` variable using jq:

```
binding_json=$(smctl get-binding example-xsuaa-binding -o json)
binding_credentials=$(echo $binding_json | jq ".items[0].credentials")
client_id=$(echo $binding_credentials | jq -r ".clientId")
```

Consuming Services with the SAP Service Manager APIs

Prerequisites

- You have obtained an API access token. For more information, see [Accessing the APIs \[page 59\]](#).
 - You have downloaded the command-line tool and library for transferring data with URLs (curl).
 - You have a command-line JSON processor at your disposal, for example jq.
1. Use the `Get All Service Offerings` API to get the list of all service offerings and retrieve the ID of the xsuaa service:

```
xsuaa_service_id=$(curl "https://service-manager.cfapps.<region domain>/v1/
service_offerings" \
    -H "Authorization: Bearer <access token>" \
    -G --data-urlencode "fieldQuery=name eq 'xsuaa'" \
    | jq -r ".items[0].id")
```

2. Use the `Get All Service Plans` API to retrieve the ID of the application service plan:

```
application_plan_id=$(curl "https://service-manager.cfapps.<region domain>/v1/
service_plans" \
    -H "Authorization: Bearer <access token>" \
    -G --data-urlencode "fieldQuery=name eq 'application'
and service_offering_id eq '$xsuaa_service_id'" \
    | jq -r ".items[0].id")
```

3. Create the service instance of the xsuaa service using the `Create a Service Instance` API:

```
xsuaa_instance_id=$(curl -X POST "https://service-manager.cfapps.<region
domain>/v1/service_instances?async=false" \
    -H "Authorization: Bearer <access token>" \
    -d '{"name": "example-xsuaa-instance", "service_plan_id":
"$application_plan_id"}' \
    | jq -r ".id")
```

The name of the new instance is `example-xsuaa-instance`, and the `async=false` setting indicates that the command output becomes available only once the instance is ready to be used.

4. Create a binding to access the service instance you created in the previous step by calling the `Create a Service Binding` API:

```
curl -X POST "https://service-manager.cfapps.<region domain>/v1/
service_bindings?async=false" \
    -H "Authorization: Bearer <access token>" \
    -d '{"name": "example-xsuaa-binding", "service_instance_id":
"$xsuaa_instance_id"}'
```

The name of the new binding is `example-xsuaa-binding`, and the output includes the `credentials` section that contains the access URLs and authorization details required to use the xsuaa service.

→ Tip

Visit [SAP Business Accelerator Hub](#) to view more details about the mentioned SAP Service Manager APIs.

5. Inject service access details to your application:

i Note

While the method to extract the relevant access and authorization details from the binding credentials is universal for all environments, the way to inject those details to an application is specific to the environment in which the application is deployed.

In this example, we extract the `clientId` field from the binding credentials, and store it in the `client_id` variable using `jq`:

```
binding_json=$(curl "https://service-manager.cfapps.<region domain>/v1/
service_bindings" \
  -G --data-urlencode "fieldQuery=name eq 'example-xsuaa-
binding'" \
  -H "Authorization: Bearer <access token>")
binding_credentials=$(echo $binding_json | jq ".items[0].credentials")
client_id=$(echo $binding_credentials | jq -r ".clientId")
```

You can now consume the xsuaa service in the environment of your choice.

Parent topic: [Consuming SAP BTP Services from Various Environments \[page 73\]](#)

Related Information

[Consuming SAP BTP Services from the Cloud Foundry Environment \[page 73\]](#)

[Consuming SAP BTP Services from the Neo Environment \[page 74\]](#)

[Consuming SAP BTP Services from the Kyma Environment \[page 74\]](#)

[Consuming SAP BTP Services from Kubernetes \[page 78\]](#)

1.2.5 Consuming SAP BTP Services from Kubernetes

Kubernetes, also known as K8s, is an open-source environment for automating deployment, scaling, and management of containerized applications.

There are two ways to consume Kubernetes services with Service Manager:

- [Consuming SAP BTP Services in Kubernetes with SAP Service Manager Broker Proxy \(Service Catalog\) \[page 79\]](#)
- [Consuming SAP BTP Services in Kubernetes with SAP BTP Service Operator \[page 82\]](#)

i Note

The first option is no longer supported. While you can still use it, we strongly recommend that you use the second option or switch to it if you are currently using the Service Manager Broker Proxy. See [Migrating from svcat to SAP BTP Service Operator \[page 86\]](#)

Parent topic: [Consuming SAP BTP Services from Various Environments \[page 73\]](#)

Related Information

[Consuming SAP BTP Services from the Cloud Foundry Environment \[page 73\]](#)

[Consuming SAP BTP Services from the Neo Environment \[page 74\]](#)

[Consuming SAP BTP Services from the Kyma Environment \[page 74\]](#)

[Consuming Services in Other Environments Using the SAP Service Manager Instances \[page 75\]](#)

1.2.5.1 Prerequisites

- You have a Kubernetes cluster and have downloaded the Kubeconfig file:
 - for [Google Cloud Platform](#) ,
 - for [Amazon EKS](#) ,
 - for [Azure](#) ,
 - for [Alibaba Cloud](#) .
- You've set your `<KUBECONFIG>` environment variable to instruct kubectl how to connect to your cluster:

```
echo "export KUBECONFIG='<full path to your kubeconfig file>.yaml'" >>
~/.bashrc
```

- kubectl v1.7 or higher, see <https://kubernetes.io/docs/tasks/tools/install-kubectl/> .
- Service Manager Control (SMCTL) CLI v1.10.1, see [Service Manager CLI Releases](#) ,
- Helm (a package manager for Kubernetes) v3.1.2, see [Helm Installation Instructions](#) .
- You have subscribed to the SAP Service Manager, see [Subscribing to SAP Service Manager \[Feature Set A\] \[page 21\]](#).
- You have assigned the *Subaccount Service Administrator role collection*, see [Assign the Subaccount Service Administrator Collection \[page 22\]](#).
- You have logged in to the SAP Service Manager, see [Logging in to SAP Service Manager \[page 23\]](#).

1.2.5.2 Consuming SAP BTP Services in Kubernetes with SAP Service Manager Broker Proxy (Service Catalog)

⚠ Caution

This consuming option is out-of-date. We strongly recommend that you [consume SAP BTP services in Kubernetes with SAP BTP service operator](#).

If you're already using SAP Service Manager Broker Proxy (Service Catalog), then first get familiar with how to [migrate from svcat to SAP BTP service operator](#).

1.2.5.2.1 Cluster Configuration

Procedure

1.  Caution

This consuming option is out-of-date. We strongly recommend that you [consume SAP BTP services in Kubernetes with SAP BTP service operator](#).

If you're already using SAP Service Manager Broker Proxy (Service Catalog), then first get familiar with how to [migrate from svcat to SAP BTP service operator](#).

Register a subaccount-scoped cluster.

```
smctl register-platform <platform name> kubernetes
```

You can choose any arbitrary name for the technical name of the cluster as long as it is unique per region.

The call above will return a similar output:

ID	Description	Name	Created	Type	Updated
Labels				Username	Password
<platform id>	<platform name>	kubernetes	2019-11-26T12:54:28.040401Z	admin	admin

- The value of *name* must be unique in the region.
 - *Username* and *Password* are the credentials that the SAP Service Manager broker proxy uses to communicate with SAP Service Manager. The credentials need to be provided in the helm chart for installing the SAP Service Manager broker proxy.
2. Install the service catalog in your Kubernetes cluster.
- a. Add the service-catalog Helm repository to your machine:

```
helm repo add svc-cat https://svc-catalog-charts.storage.googleapis.com
```

- b. Create the catalog namespace:

```
kubectl create namespace catalog
```

- c. Install the Service Catalog in the catalog namespace:

```
helm install catalog svc-cat/catalog --namespace catalog --version 0.3.0
```

See [Install Service Catalog using Helm](#) 

iNote

Svcat v0.3.0 is required for compatibility with Kubernetes clusters v1.17.4.

3. Install the SAP Service Manager broker proxy in your Kubernetes cluster. See [service-broker-proxy-k8s](#) .
 - a. Add the Peripli Helm repository to your machine:

```
helm repo add peripli 'https://peripli.github.io'
```

- b. Create the service-broker-proxy namespace:

```
kubectl create namespace service-broker-proxy
```

- c. Install the SAP Service Manager Broker Proxy in the service-broker-proxy namespace:

```
helm install service-broker-proxy peripli/service-broker-proxy-k8s \
  --namespace service-broker-proxy \
  --version <VERSION> \
  --set config.sm.url=<SM_URL> \
  --set sm.user=<USER> \
  --set sm.password=<PASSWORD>
```

- Replace the SM_URL with the URL of the SAP Service Manager.
Syntax: `https://service-manager.cfapps.<landscape domain>` (to find the landscape domain of the SAP Service Manager, see [Regions and API Endpoints Available for the Cloud Foundry Environment](#)).
- Replace the USER and PASSWORD with the credentials obtained in Step 2.
- Replace the VERSION with the required version as listed on [Releases](#) . It is recommended to use v0.7.0.

1.2.5.2.2 Working with Service Catalog

Procedure

1.  Caution

This consuming option is out-of-date. We strongly recommend that you [consume SAP BTP services in Kubernetes with SAP BTP service operator](#).

If you're already using SAP Service Manager Broker Proxy (Service Catalog), then first get familiar with how to [migrate from svcat to SAP BTP service operator](#).

To use the Service Catalog and get an access to the marketplace, install the Service Catalog CLI by executing the following commands:

- For Mac OS:

```
curl -sLO https://download.svcat.sh/cli/latest/darwin/amd64/svcat
```

```
chmod +x ./svcat
```

```
mv ./svcat /usr/local/bin/
```

```
svcat version --client
```

- For Windows OS:

```
iwr 'https://download.svcat.sh/cli/latest/windows/amd64/svcat.exe'  
-UseBasicParsing -OutFile svcat.exe
```

```
mkdir -f ~\bin
```

```
Move-Item -Path svcat.exe -Destination ~\bin  
Move-Item -Path svcat.exe -Destination ~\bin
```

```
$env:PATH += ";${pwd}\bin"
```

```
svcat version --client
```

See [Install the Service Catalog CLI](#) .

2. Use the Service Catalog CLI to present the list of all the SAP BTP services available for the Kubernetes environment:

```
svcat marketplace
```

or

```
svcat mp
```

3. Create an instance of any service, by specifying its name and plan:

```
svcat provision <Enter a name for the instance> --class xsuaa --plan  
application
```

4. Verify the service instance was created successfully:

```
svcat get instances
```

5. Delete the service instance:

```
svcat deprovision INSTANCE-NAME
```

1.2.5.3 Consuming SAP BTP Services in Kubernetes with SAP BTP Service Operator

Learn how to consume SAP Business Technology Platform (SAP BTP) services using the SAP Service Manager service (technical name: service-manager) from a Kubernetes cluster not managed by SAP.

Context

You consume services from an SAP BTP global account and a subaccount.

As subaccounts are bound to specific regions, choose or create a subaccount located close to your Kubernetes cluster to prevent latency drawbacks.

You manage SAP BTP services with the SAP BTP service operator from your Kubernetes cluster in a Kubernetes-native way. The SAP BTP service operator is based on the Kubernetes CustomResourceDefinition API with which you can manage SAP BTP services from the cluster by calling the Kubernetes APIs.

For more information, see [SAP BTP Service Operator](#) .

Related Information

<https://developers.sap.com/tutorials/btp-hyperscaler-extension.html>

1.2.5.3.1 Setup

Procedure

1. Install cert-manager.
Cert-manager is used to set up the certificates needed for internal communication between the Kubernetes API server and the deployment of the SAP BTP service operator in your cluster.
For more information, see [Kubernetes cert-manager](#) .
2. Obtain the access credentials for the SAP BTP service operator:
 1. Using the SAP BTP cockpit or Service Manager Control (SMCTL) command-line interface, create an instance of the SAP Service Manager (technical name: `service-manager`) with the plan: `service-operator-access`.

Note

If you can't see the needed plan, you need to entitle your subaccount to use SAP Service Manager.
For more information about how to entitle a service to a subaccount, see [Configure Entitlements and Quotas for Subaccounts](#).

For more information about creating service instances, see:

- [Creating Instances in Other Environments \[page 13\]](#)
 - [provision \[page 39\]](#) (SMCTL command name)
2. Create a binding to the created service instance.
For more information about creating service bindings, see:
 - [Creating Service Bindings in Other Environments \[page 18\]](#)
 - [bind \[page 45\]](#)
 3. Retrieve the generated access credentials from the created binding object.
The example of the binding object created with the default credentials type:

```
{
  "clientid": "xxxxxxx",
  "clientsecret": "xxxxxxx",
  "url": "https://mysubaccount.authentication.eu10.hana.ondemand.com",
  "xsappname": "<name>",
```

```
}
  "sm_url": "<service_manager_URL>"
}
```

The example of the binding object created with the X.509 credentials type:

```
{
  "clientid": "xxxxxxx",
  "certificate": "-----BEGIN CERTIFICATE-----XXX-----END
CERTIFICATE",
  "key": "-----BEGIN RSA PRIVATE KEY-----XXX-----END RSA PRIVATE
KEY-----",
  "certurl": "<certificate_URL>",
  "xsappname": "<name>",
  "sm_url": "<service_manager_URL>"
}
```

3. Deploy the SAP BTP service operator in the cluster using the obtained access credentials.

The example of the deployment that uses the default access credentials type:


```
helm upgrade --install sapbtp-operator https://github.com/SAP/sap-btp-service-
operator/releases/download/<release>/sapbtp-operator-<release>.tgz \
--create-namespace \
--namespace=sapbtp-operator \
--set manager.secret.clientid=<clientid> \
--set manager.secret.clientsecret=<clientsecret> \
--set manager.secret.sm_url=<sm_url> \
--set manager.secret.tokenurl=<url>
```

The example of the deployment that uses the X.509 access credentials type:

```
helm upgrade --install sap-btp-operator https://github.com/SAP/sap-btp-
service-operator/releases/download/<release>/sap-btp-operator-<release>.tgz \
--create-namespace \
--namespace=sap-btp-operator \
--set manager.secret.clientid=<clientid> \
--set manager.secret.tls.crt="$(cat /path/to/cert)" \
--set manager.secret.tls.key="$(cat /path/to/key)" \
--set manager.secret.sm_url=<sm_url> \
--set manager.secret.tokenurl=<certurl>
```

For the list of the available SAP BTP service operator releases, see [Available Releases](#) .

Note

To learn more about how to consume SAP BTP services from any hyperscaler, see [Consume SAP BTP Services from Any Hyperscaler](#) .

1.2.5.3.2 Working with SAP BTP Service Operator

Use the SAP BTP service operator to consume SAP BTP services from your Kubernetes cluster.

Context

To consume an SAP BTP service, you create an instance of that service, and then you create a binding so that your Kubernetes-native applications can get access credentials to the instance you created.

Procedure

Creating a Service Instance

1. Create a ServiceInstance custom resource file with the following structure:

```
apiVersion: services.cloud.sap.com/v1
kind: ServiceInstance
metadata:
  name: my-service-instance
spec:
  serviceOfferingName: <offering>
  servicePlanName: <plan>
  externalName: my-service-instance-external
  parameters:
    key1: val1
    key2: val2
```

where:

- **<offering>** is the name of the SAP BTP service whose instance you're creating. To learn more about viewing and managing the available services for your subaccount in the SAP BTP cockpit, see [View and Manage Services from the Service Marketplace \[page 8\]](#).
 - **<plan>** is the plan of the selected service offering for which you're creating an instance.
2. Apply the custom resource file from step 1 in your cluster by running the following command:

```
kubectl apply -f path/to/my-service-instance.yaml
```

3. Check that the status of the service instance in your cluster is **Created** by running the following command:

```
kubectl get serviceinstances
NAME                STATUS    AGE
my-service-instance Created   19s
```

Creating a Service Binding

1. Create a ServiceBinding custom resource file and set the `serviceInstanceName` field value to the name of the ServiceInstance custom resource file you created in the first section:

```
apiVersion: services.cloud.sap.com/v1
kind: ServiceBinding
metadata:
  name: my-binding
spec:
  serviceInstanceName: my-service-instance
```

2. Apply the custom resource file from step 1 in your cluster by running the following `kubectl` command:

```
kubectl apply -f path/to/my-binding.yaml
```

3. Check that the status of the service binding in your cluster is **Created** by running the following command:

```
kubectl get servicebindings
NAME      INSTANCE                STATUS    AGE
my-binding  my-service-instance    Created   16
```

4. Check that a secret with the same name as the name of your binding is created.

The secret contains credentials applications in your cluster can use to access the service instance:

```
kubectl get secrets
NAME      TYPE      DATA   AGE
my-binding  Opaque    5       32s
```

For more information about how to use the generated credentials from your applications in Kubernetes cluster, see [Using Secrets](#).

1.2.5.4 Migrating from svcat to SAP BTP Service Operator

Learn how to migrate a registered Kubernetes platform, based on the Kubernetes Service Catalog (svcat) and SAP Service Manager agent, together with its content, to an SAP BTP service operator-based platform.

Context

SAP BTP service operator enables you to provision and consume SAP BTP offerings from Kubernetes cluster in a Kubernetes-native way, based on the Kubernetes Operator pattern.

Prerequisites

- You have Service Manager Control (SMCTL) Command-Line tool. See [Using the Service Manager Control \(SMCTL\) Command-Line Tool \[page 20\]](#).
- You must be an SAP BTP subaccount admin.
- Perform steps 1 and 2 described in the [Setup \[page 83\]](#) topic.

Procedure

The procedure consists of two main groups of steps; first, you set up the service operator and the migration command line interface in the same cluster in which your Kubernetes service-catalog content is located, then you perform the migration.

Setup

1. Deploy the SAP BTP service operator by executing the following command:

```
helm upgrade --install sap-btp-operator https://github.com/SAP/sap-btp-
service-operator/releases/download/<release>/sap-btp-operator-<release>.tgz \
  --create-namespace
  --namespace=sap-btp-operator
  --set manager.secret.clientid=<clientid>
  --set manager.secret.clientsecret=<clientsecret>
  --set manager.secret.url=<sm_url>
  --set manager.secret.tokenurl=<url>
```

```
--set cluster.id=<clusterID>
```

i Note

For `<clusterID>`, specify the ID of the cluster. To find it, run the following command:

```
kubectl get configmap -n catalog cluster-info -o yaml
```

and extract `id` from the output.

Output example:

```
apiVersion: v1
data:
id: ab7fa5e9-5cc3-468f-ab4d-143458785d07
kind: ConfigMap
metadata:
.
.
```

2. Download and install SAP BTP service operator CLI in one of the following ways:

- Manual Installation:

1. Get the CLI needed to perform the service operator migration by running the following command:

```
go get github.com/SAP/sap-btp-service-operator-migration
```

2. Install the CLI:

```
go install github.com/SAP/sap-btp-service-operator-migration
```

3. Rename the CLI binary:

```
mv $GOPATH/bin/sap-btp-service-operator-migration $GOPATH/bin/migrate
```

- Automatic Installation:

Download the latest release of the CLI. For more information, see [SAP BTP Service Operator CLI Releases](#).

You need the CLI to perform the migration.

CLI Overview

```
Tool used to migrate content from SVCAT to SAP BTP Service Operator-based
platform.
Usage:
migrate [flags]
migrate [command]
Available Commands:
dry-run    Run migration in dry-run mode
help       Help about any command
run        Run migration process
version    Prints migrate version
Flags:
-c, --config string      Config file (default is $HOME/.migrate/config.json)
-h, --help               Help command for the migrate
-k, --kubeconfig string  Absolute path to the kubeconfig file (default
$HOME/.kube/config)
-n, --namespace string   Specify the namespace to find operator secret
(default sap-btp-operator)
```

Migration

1. Prepare your platform for migration by executing the following command:

```
smctl curl -X PUT -d '{"sourcePlatformID": ":platformID"}' /v1/migrate/  
service_operator/:instanceID
```

Where the parameter values are as following:

- `platformID` is the ID that was generated when you registered a subaccount-scoped cluster in the step 1 of the Cluster Configuration topic. See [Cluster Configuration \[page 80\]](#).
 - `instanceID` is the ID of the `service-operator-access` instance created in the step 1 of the Setup subsection.
2. Execute the migration by running the following command:

```
migrate run
```

→ Tip

You can perform a dry run before you execute the migration by running the command:

```
migrate dry-run
```

Dry run is useful if you wish to execute the scan and validation steps described in the Migration Script Example section below without performing the actual migration.

At the end of the run, summary including all encountered errors is shown.

This way, you can decide whether to continue with the migration or first fix the issues.

i Note

Once the actual migration process has been initiated, the platform is suspended, and you can't create, update, or delete its service instances and service bindings.

The process is reversible for as long as the actual migration of the resources doesn't start (described below in the part 3 of the migration script example).

To cancel the migration, execute the following command:

```
smctl curl -X DELETE -d '{"sourcePlatformID": ":platformID"}' /v1/migrate/  
service_operator/:instanceID
```

Migration Script Example

1. The script first scans all service instances and service bindings that are managed in your cluster by Kubernetes Service Catalog (svcat), and verifies whether they're also maintained in SAP BTP. Migration isn't performed on those instances and bindings that aren't found in SAP BTP.

```
migrate run  
  
*** Fetched 2 instances from SM
```



```

*** Fetched 1 bindings from SM
*** Fetched 5 svcat instances from cluster
*** Fetched 2 svcat bindings from cluster
*** Preparing resources

svcat instance name 'some_instance_name_1' id 'XXX-6134-4c89-bff5-YYY'
(some_instance_name_1) not found in SM, skipping it...
svcat instance name 'some_instance_name_2' id 'XXX-cae6-4e23-9e8a-YYY'
(some_instance_name_2) not found in SM, skipping it...
svcat instance name 'some_instance_name_3' id 'XXX-dc1d-49d1-86c0-YYY'
(some_instance_name_3) not found in SM, skipping it...
svcat binding name 'some_binding_name_1' id 'XXX-5226-42cc-81e5-YYY'
(some_binding_name_1) not found in SM, skipping it...

*** found 2 instances and 1 bindings to migrate

```

2. Before the actual migration starts, the script also validates whether all resources are migratable.

Note

If there's an issue with one or more resources, the process stops.

```

svcat instance 'some_instance_name_4' in namespace 'namespace_name' was
validated successfully
svcat instance 'some_instance_name_5' in namespace 'namespace_name'
was validated successfully
svcat binding 'some_binding_name_2' in namespace 'namespace_name' was
validated successfully

*** Validation completed successfully

```

3. After all resources were validated successfully, the actual migration starts. Each service instance and binding is removed from the Service Catalog and added to the SAP BTP service operator:

```

migrating service instance 'some_instance_name_4' in namespace
'namespace_name' (smID: 'XXX-3d1f-40db-8cac-YYY')
deleting svcat resource type 'serviceinstances' named
'some_instance_name_4' in namespace 'namespace_name'
migrating service instance 'some_instance_name_5' in namespace
'namespace_name' (smID: 'XXX-0f94-4fde-b524-YYY')
deleting svcat resource type 'serviceinstances' named
'some_instance_name_5' in namespace 'namespace_name'
migrating service binding 'some_binding_name_2' in namespace
'namespace_name' (smID: 'XXX-fc36-4d50-a925-YYY')
deleting svcat resource type 'servicebindings' named
'some_binding_name_2' in namespace 'namespace_name'

*** Migration completed successfully

```

Note



Once the migration process has been completed, the Kubernetes platform, based on the Kubernetes Service Catalog (svcat) and SAP Service Manager agent is no longer usable.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.

About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
 - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
 - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.

The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

© 2023 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see <https://www.sap.com/about/legal/trademark.html> for additional trademark information and notices.