PUBLIC
2021-05-27

SAP Credential Store

THE BEST RUN SAP

Content

1 SAP Credential Store

Get Started

Concepts

Operations

Resources

1.1 What Is SAP Credential Store?

Store and retrieve credentials such as cryptographic keys and passwords.

SAP Credential Store service provides a repository for passwords and keys for applications that are running on SAP BTP. It enables the applications to retrieve credentials and use them for authentication to external services, or to perform cryptographic operations and TLS communication. SAP Credential Store is exposed to the applications via a REST API.

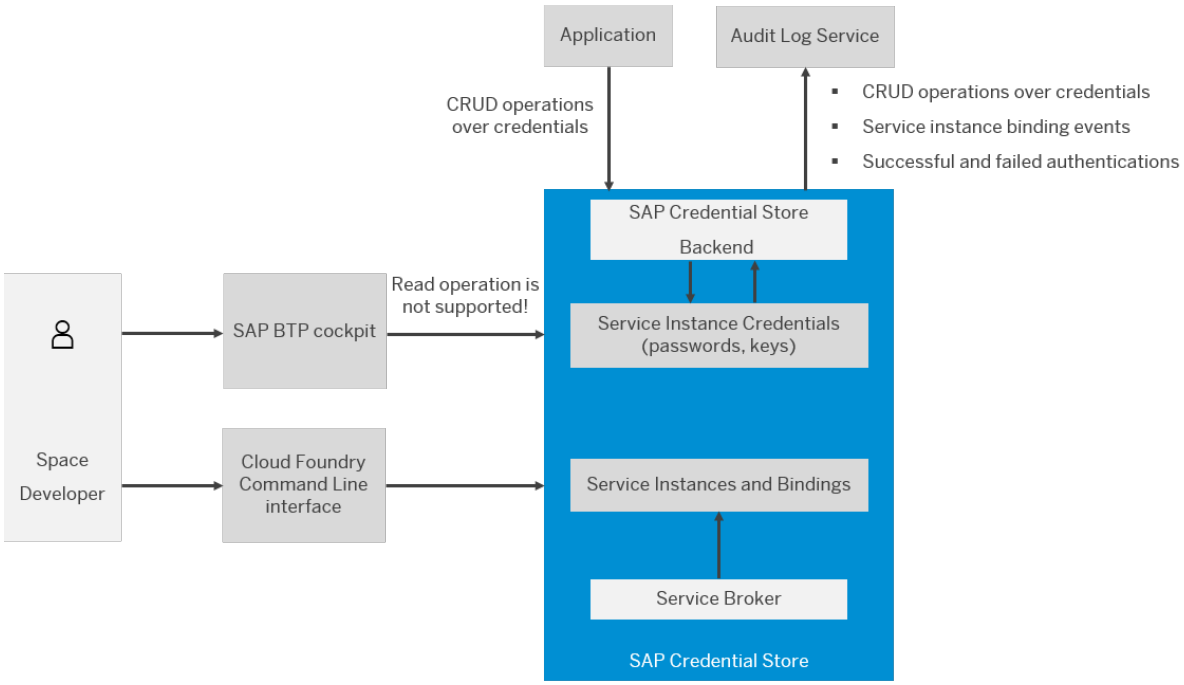
Environment

SAP Credential Store runs in the SAP BTP, Cloud Foundry environment.

Features

Store passwords and keys	Use the repository to store cryptographic keys and passwords for business applications in the Cloud Foundry environment.
Retrieve credentials	Retrieve passwords and keys to use them for authentication to external services.
Manage service instances	Create and bind service instances by using SAP BTP cockpit or Cloud Foundry Command Line interface (CF CLI).
Create service keys	Create a service key if you need to use a service instance from an external application or an application deployed in another space.

Overview Graphic



Prerequisites

- You need to have a space in a Cloud Foundry environment global account.
- Your global account must support the consumption-based commercial model.
- Request a quota for the SAP Credential Store and assign it to subaccounts.

For more information, see Initial Setup [page 8].

The following plans and metrics are available to you:

Plan	Bindings	Credentials	Password Size	Key Size	Total Size	API Calls per Second	Namespaces
standard	100	100,000	4 KB	32 KB	100 MB	25	10,000
proxy*	100	100,000	4 KB	32 KB	100 MB	25	10,000

*A **proxy** plan inherits the standard plan along with its metrics. Also, the proxy plan is only applied to scenarios that require service instance sharing. For more information, see: Share, Unshare, and Authorize a Service Instance [page 17]

> **!** Restriction
>
> If you create multiple proxy instances related to a single standard instance, then all these proxy instances together will be allowed to create up to 100,000 credentials. The same rule applies to all other metrics – they can altogether make up to 25 calls per second, or bind up to 100 applications, and so on.

Tools

You can access the *Credential Store* tile in the SAP BTP cockpit, in your Cloud Foundry environment space.

Restrictions

You can use only **1** service instance per space.

Regions

The service is available in the following Cloud Foundry environment regions:

- Australia (Sydney) – *running on AWS*
- Brazil (São Paulo) – *running on AWS*
- Canada (Montreal) – *running on AWS*
- China (Shanghai) – *running on Alibaba*
- Europe (Frankfurt) – *running on AWS*
- Europe (Netherlands) – *running on Azure*
- Japan (Tokyo) – *running on Azure*
- Japan (Tokyo) – *running on AWS*
- Singapore – *running on AWS*
- Singapore – *running on Azure*
- US East (VA) – *running on AWS*
- US East (VA) – *running on Azure*

- US Central (IA) – *running on GCP*
- US West (WA) – *running on Azure*

For more information, see:

Cloud Foundry Regions and Endpoints

Discovery Center: SAP Credential Store

Trial Scope

SAP Credential Store is available in your SAP Business Technology Platform trial account, Cloud Foundry environment. See also:

- Trial Accounts – for general information on SAP BTP trial
- Get a Trial Account – for a detailed explanation on how to register for a free trial account.

The following parameters are available to you when working with SAP Credential Store in your free trial account:

Plan	Bindings	Credentials	Password Size	Key Size	Total Size	API Calls per Second	Namespaces
trial	3	10	4 KB	32 KB	0,1 MB	3	10

1.2 What's New for SAP Credential Store

To check the latest release notes for the SAP Credential Store, go to SAP BTP: What's New (Credential Store).

(Optional) You can change the default date filter (*From* – *To*) in order to see an extended or a narrowed range of release notes for the SAP Credential Store.

07 October 2020

New

Trial Scope

You can now test for free a trial version of the SAP Credential Store service.

To learn more, see What Is SAP Credential Store? [page 3] → section **Trial Scope**.

10 August 2020

New

Sharing and unsharing service instances

You can now use a **proxy** plan to share and unshare service instances in the same or in a different region. You can also control the permissions of proxy instances so that they can only have limited access to credentials in a particular namespace.

To learn how, see:

- What Is SAP Credential Store? [page 3] → section **Prerequisites**
- Share, Unshare, and Authorize a Service Instance [page 17]

09 April 2020

New

Creating and managing namespaces and credentials

You can now create, edit and delete credentials and namespaces in the SAP BTP cockpit UI. To learn how, see:

- Create and Delete a Namespace [Feature Set A] [page 21]
- Create, Edit, and Delete a Credential [Feature Set A] [page 23]

New

Filtering namespaces and credentials

If you have created more than 10 namespaces, or more than 100 credentials of a single type, they will not all be displayed in the relevant table. Thus, you can use filtering to find a particular one.

To learn how, see: Set Filters for Credentials and Namespaces [Feature Set A] [page 24]

12 September 2019

New

More regions available

The SAP Credential Store is now available in Canada, Japan, US West, Netherlands, China, and Singapore.

To see the full list of regions, see What Is SAP Credential Store? [page 3] → **Regions**.

19 February 2019

New

SAP Credential Store

SAP Credential Store is a service that provides a secure repository for passwords and keys to the applications running on SAP BTP, Cloud Foundry environment. To learn more, see:

- What Is SAP Credential Store? [page 3]
- Initial Setup [page 8]
- REST API [page 25]
- Security [page 36]

1.3 Initial Setup

Prerequisites

- You need to have a space in a Cloud Foundry environment global account.
- Your global account must support the consumption-based commercial model. To learn more, see the consumption-based relevant details on page: Commercial Models and Metering
- Request a quota for the SAP Credential Store and assign it to subaccounts.

Context

To enable the SAP Credential Store for your account, follow the steps below.

Procedure

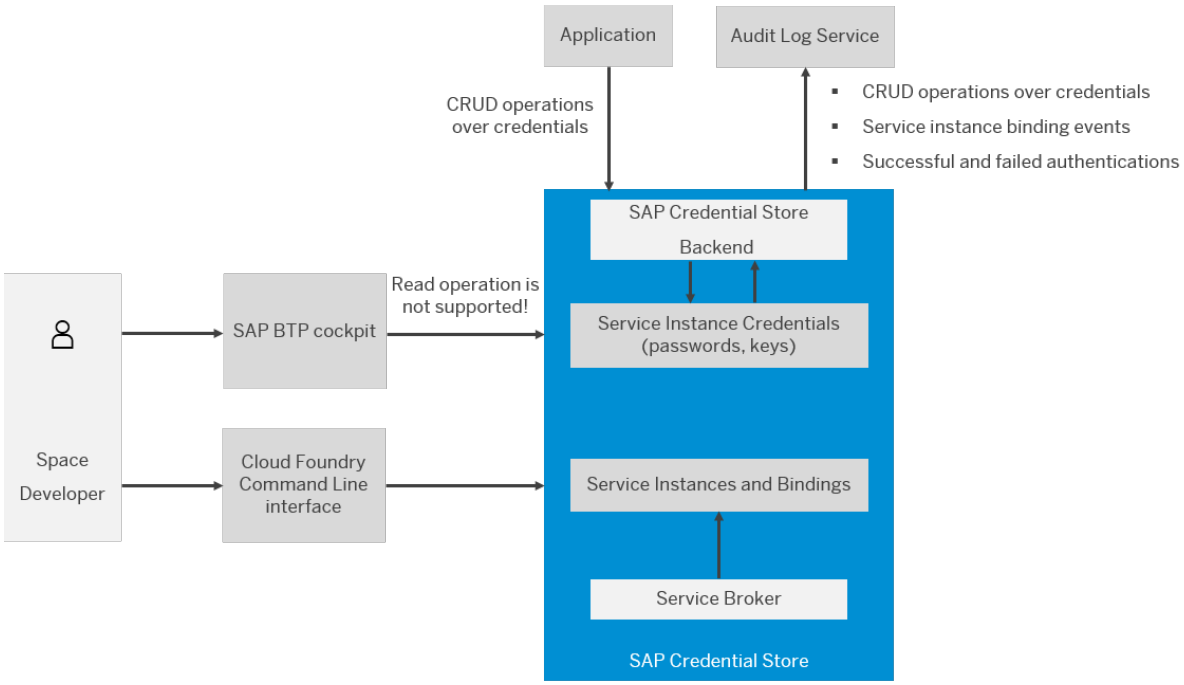
1. Go to SAP BTP cockpit and open your account. See: Managing Subaccounts Using the Cockpit
2. Choose *Entitlements* to edit the service parameters entitled to this account.
3. Now go to your space and choose *Service Marketplace*.
4. Choose the *Credential Store* tile.
5. Take a look at the service overview. If you need, you can add new service instances.

Related Information

Operations [page 13]

1.4 Concepts

This section describes basic concepts that will help you familiarize yourself with the SAP Credential Store service.



SAP BTP cockpit

You can perform the following operations through SAP BTP cockpit:

- Create, update (edit), and delete credentials
- Create and bind service instances
- Create and delete namespaces
- Create, download, and delete service keys
- Set filters for credentials and namespaces

> **! Restriction**
>
> You cannot read credential values.

Cloud Foundry Command Line interface

You can perform the following operations through CF CLI:

- Create and bind service instances
- Create, download, and delete service keys

> ! Restriction
>
> You cannot operate with credentials.

Related Information

Service Features [page 10]
Authorization of Bindings [page 12]

1.4.1 Service Features

Namespaces

The data in the SAP Credential Store is logically isolated using namespaces. A namespace can correspond to a customer, a subaccount (tenant) or anything else specific to an application. Each credential operation is executed in the context of a namespace.

Allowed characters in a namespace:

- letters
- digits
- underscore (_)
- dash (-)
- dot (.)
- colon (:)
- exclamation mark (!)
- tilde (~)

> i Note
>
> A namespace can consist of up to **100** chars.

Credentials

You can store two types of credentials in a namespace:

- Passwords
- Keys

Each credential has a name, a value and a set of optional attributes: *metadata*, *format*, *username*

Credential	Name	Value	Metadata	Format	Username
Password	✓	✓	✓		✓
Key	✓	✓	✓	✓	✓

Depending on the credential type, the value can be either text (password) or binary (key). In the table below are listed the details about each credential attribute.

Attribute	Type	Max Size
`name`	text	255 chars
`value` (password)	text	4096 chars
`value` (key)	binary	32 KB
`metadata`	text	10,000 chars
`format`	text	255 chars
`username`	text	1024 chars

Credential Operations

The SAP Credential Store supports several operations over a credential.

Operation	Method	Header
read	GET	
read if changed	GET	If-None-Match: <id>
create	POST	If-None-Match: *
create or update	POST	
update	POST	If-Match: *
update if not changed	POST	If-Match: <id>
delete	DELETE	

Some operations are called using the same HTTP method. In this case, the HTTP headers "*If-Match*" or "*If-None-Match*" are used to define the exact operation. For more information, see *REST API definition*.

1.4.2 Authorization of Bindings

If a service instance is shared among multiple applications, you can define different access permissions for each service binding. The permissions are defined when a service binding or a service key is created. For more information see:

- Bind a Service Instance [page 15]
- Create, Download, and Delete a Service Key [page 20]

The permissions configuration is a JSON document in the following format:

```
{
  "authorization": {
    "default_permissions": [
      // list of default credential permissions
    ],
    "namespace_permissions": {
      "<namespace_1>": [
        // list of credential permissions for namespace <namespace_1>
      ],
      "<namespace_2>": [
        // list of credential permissions for namespace <namespace_2>
      ],
      ...
      "<namespace_n>": [
        // list of credential permissions for namespace <namespace_n>
      ]
    }
  }
}
```

The same permissions configuration is used also when a service instance is shared. See: Share, Unshare, and Authorize a Service Instance [page 17]

Credential Permissions

- *create*: allows to create a credential in a namespace
- *read*: allows to read any credential in a namespace
- *update*: allows to update any credential in a namespace
- *delete*: allows to delete any credential in a namespace
- *list*: allows to list the credentials (passwords and keys) in a namespace

Examples

> **Example**
>
> Allows to create, read, update, and delete credentials in namespace *namespace1* and to list credentials in any other namespace.

Note: This configuration does not allow listing of credentials in *namespace1*.

```
{
  "authorization": {
    "default_permissions": [
      "list"
    ],
    "namespace_permissions": {
      "namespace1": [
        "create",
        "read",
        "update",
        "delete"
      ]
    }
  }
}
```

Example

Allows to list credentials in any namespace.

```
{
  "authorization": {
    "default_permissions": [
      "list"
    ]
  }
}
```

Example

Allows to create, read, update, delete, and list credentials in namespace *namespace1*.

```
{
  "authorization": {
    "namespace_permissions": {
      "namespace1": [
        "create",
        "read",
        "update",
        "delete",
        "list"
      ]
    }
  }
}
```

1.5 Operations

This section describes the operations you can perform with the SAP Credential Store service.

Related Information

Create a Service Instance [page 14]
Bind a Service Instance [page 15]
Create, Download, and Delete a Service Key [page 20]
Share, Unshare, and Authorize a Service Instance [page 17]

1.5.1 Create a Service Instance

Prerequisites

- You have a space in a Cloud Foundry environment global account.
- Your global account supports the consumption-based commercial model.
- You have requested a quota for SAP Credential Store and assigned it to subaccounts.

For more information, see: Initial Setup [page 8]

Context

Create a service instance. You can do this by using the Cloud Foundry environment command line interface. For more information, see: https://docs.cloudfoundry.org/devguide/services/managing-services.html

Console command example:

Sample Code

```
cf create-service credstore standard my-credstore
```

! Restriction

You can use only **1** service instance per space.

Alternatively, you can create a service instance through the cockpit.

Procedure

1. Go to SAP BTP cockpit and open your account. See: Managing Subaccounts Using the Cockpit
2. Choose the *Credential Store* tile.

3. Choose *Instances* and then *New Instance*.
4. Select service plan **standard** from the drop-down box.

> **i** Note
>
> To have this option available, you need to manage an entitlement for the **standard** service plan. See: Configure Entitlements and Quotas for Subaccounts

5. (Optional) Bind the service instance to an application.
6. Choose *Confirm*. The service instance is created.

1.5.2 Bind a Service Instance

Prerequisites

- You have a space in a Cloud Foundry environment global account.
- Your global account supports the consumption-based commercial model.
- You have requested a quota for SAP Credential Store and assigned it to subaccounts.

For more information, see: Initial Setup [page 8]

Context

Bind a service instance to an application, or create service keys. You can do this by using the Cloud Foundry environment command line interface. For more information, see: https://docs.cloudfoundry.org/devguide/services/managing-services.html

Console command example:

> Sample Code

```
cf bind-service myapp my-credstore
```

Alternatively, you can bind a service instance through the cockpit.

Procedure

1. Go to SAP BTP cockpit and open your account. See: Managing Subaccounts Using the Cockpit
2. Choose the *Credential Store* tile.

3. Navigate to the corresponding service instance.
4. Choose *Bind Instance*.
5. Select the application name.
6. (Optional)Define parameters for this binding.

   For example: **authorization**. If a service instance is shared among multiple applications, you can define different access permissions for each application (binding). For example, an application may have full access (*list*, *read*, *write*) to the credentials stored in "`namespace1`" but only *list* permissions for the other namespaces.

   ### Sample Code

   ```
   {
       "authorization": {
           "default_permissions": [
               "list"
           ],
           "namespace_permissions": {
               "namespace1": [
                   "list",
                   "write",
                   "read"
               ]
           }
       }
   }
   ```

   ### Note

   At runtime, the namespace specific access permissions are evaluated first. The default permissions are applied only if none of the namespace permissions matches.

7. Choose *Save*. The binding is created

Next Steps

After the binding is created, several properties are available for the application via the VCAP_SERVICES environment variable.

- **username:** this property is used for authentication. It contains the unique binding ID.
- **private_key:** this property contains the private key used to sign JWT tokens for authentication. See: Encrypting Payloads [page 29]
- **oauth_token_url:** the SAP Credential Store supports OAuth client credentials flow for authentication. This property contains the token URL of the OAuth authorization server, which accepts either signed JWT tokens or username/password and issues access tokens.
- **url:** this property contains the URL of the SAP Credential Store REST API. The URL can be accessed only with a valid access token issued by the OAuth authorization server (see property **oauth_token_url**).
- **parameters:** this property contains either the default binding options, or the custom ones defined during the binding creation. Currently, it contains only an authorization section with access permissions.

To find them in the cockpit, open *Environment Variables* and see:

Sample Code

```
{
    "oauth_token_url":
    "https://securestoreauth.cfapps.sap.hana.ondemand.com/api/v1/token",
    "private_key": "...",
    "parameters": {
        "authorization": {
            "default_permissions": [
                "list",
                "write",
                "read"
             ],
             "namespace_permissions": {
                "<namespace>": [
                    "list",
                    "write",
                    "read"
                ]
             }
         }
     },
     "url": "https://securestore.cfapps.sap.hana.ondemand.com/api/v1/
credentials",
     "username": "1234567a-1234-1a1a-9876-123456789abc"
}
```

1.5.3 Share, Unshare, and Authorize a Service Instance

The SAP Credential Store implements its own mechanism for sharing service instances, which differs from the standard mechanism supported by Cloud Foundry . The main characteristics of the mechanism supported by the SAP Credential Store are:

- Two-step process – *step 1* is executed by a Space Developer in the source space, and *step 2* is executed by a Space Developer in the target space
- Possibility to share with a target space in the same region
- Possibility to share with a target space in another region
- Possibility to define authorizations

Prerequisites

To share a service instance from one space (source space) to another space (target space), you need to know the GUID of the service instance you want to share and the GUID of the target space. You can check them using the following CF CLI commands:

Sample Code

```
cf service my-credstore --guid
cf space target-space --guid
```

Sharing a Service Instance

To illustrate these operations better, let's use the following exemplary values:

- The name of the service instance you want to share is: *my-credstore*
- The GUID of the service instance is: *aaaaaa-111111-2222-abcabcabc*
- The GUID of the target space is: *777777-00000-ccccc-11111-ddddd*
- The name of the proxy service instance you want to create is: *proxy-credstore*

> **i** Note
>
> All examples below are applicable to Microsoft Windows. If you use a command console for another OS, you might need to use a different approach to escape the double inverted commas (").

Share a service instance in the same region

1. Step 1 (source space):
   Execute a CF CLI command to update the service instance with the following parameters:

   ```
   cf update-service my-credstore -c "{\"share\":{\"spaceGuid\":\"777777-00000-
   ccccc-11111-ddddd\"}}"
   ```

2. Step 2 (target space):
   Create a SAP Credential Store proxy service instance with the following parameters:

   ```
   cf create-service credstore proxy proxy-credstore -c "{\"origin\":
   {\"serviceGuid\":\"aaaaaa-111111-2222-abcabcabc\"}}"
   ```

Share a service instance in another region

To share a service instance with a target space in another region, you have to know the landscape name. Below is the list of these mappings:

Provider	Region	Landscape
Amazon Web Services (AWS)	Australia (Sydney)	`cf-ap10`
Amazon Web Services (AWS)	Singapore	`cf-ap11`
Amazon Web Services (AWS)	Brazil (São Paulo)	`cf-br10`
Amazon Web Services (AWS)	Canada (Montreal)	`cf-ca10`
Amazon Web Services (AWS)	Europe (Frankfurt)	`cf-eu10`
Amazon Web Services (AWS)	Japan (Tokyo)	`cf-jp10`
Amazon Web Services (AWS)	US East (VA)	`cf-us10`
Microsoft Azure	Singapore	`cf-ap21`
Microsoft Azure	Europe (Netherlands)	`cf-eu20`
Microsoft Azure	Japan (Tokyo)	`cf-jp20`
Microsoft Azure	US West (WA)	`cf-us20`
Microsoft Azure	US East (VA)	`cf-us21`

Provider	Region	Landscape
Google Cloud Platform	US Central (IA)	`cf-us30`
SAP	Europe (Rot)	`cf-eu1`

Let's assume that the source space is in landscape *cf-eu10*, and the target space is in landscape *cf-us20*. Then:

1. Step 1 (source space):
   Execute CF CLI command to update the service instance with the following parameters:

   ```
   cf update-service my-credstore -c "{\"share\":{\"landscape\":\"cf-us20\",
   \"spaceGuid\":\"777777-00000-ccccc-11111-ddddd\"}}"
   ```

2. Step 2 (target space):
   Create a SAP Credential Store proxy service instance with the following parameters:

   ```
   cf create-service credstore proxy proxy-credstore -c "{\"origin\":{\"landscape
   \":\"cf-eu10\",\"serviceGuid\":\"aaaaaa-111111-2222-abcabcabc\"}}"
   ```

Authorizing a Service Instance

In case a service instance is shared with a "foreign" target space, that is, a space which is not under the control of the administrator of the source space, you may want to reduce the access permissions of the proxy service instance. This can be done in *Step 1* of sharing a service instance.

For example:

> ### Example
>
> ```
> cf update-service my-credstore -c "{\"share\":{\"spaceGuid\":\"777777-00000-
> ccccc-11111-ddddd\",\"authorization\":{\"namespace_permissions\":{\"shared-
> namespace\":[\"create\",\"delete\",\"list\",\"read\",\"update\"]}}}}"
> ```
>
> In this example, the proxy service instance will have full access to the credentials in namespace *shared-namespace* only. It won't be able to do any operations over credentials in other namespaces.

The authorization configuration follows the structure used for defining the binding authorizations. For more information, see: Authorization of Bindings [page 12]

Unsharing a Service Instance

To unshare a service instance, you need to perform an operation to update the source service instance with specific configuration. This configuration defines if all shares for all regions shall be removed, or only for a specific region, or only for a specific space in a region.

See the examples below.

Example

Unshare a service instance from all regions:

```
cf update-service my-credstore -c "{\"unshare\":{\"landscape\":\"*\"}}"
```

Example

Unshare a service instance from a specific region:

```
cf update-service my-credstore -c "{\"unshare\":{\"landscape\":\"cf-us20\"}}"
```

Example

Unshare service instance from a specific target space in a region:

```
cf update-service my-credstore -c "{\"unshare\":{\"landscape\":\"cf-us20\",
\"spaceGuid\":\"777777-00000-ccccc-11111-ddddd\"}}"
```

1.5.4 Create, Download, and Delete a Service Key

Prerequisites

- You have a space in a Cloud Foundry environment global account.
- Your global account supports the consumption-based commercial model.
- You have requested a quota for SAP Credential Store and assigned it to subaccounts.

For more information, see: Initial Setup [page 8]

Context

If you need to use a service instance from an external application, or an application deployed in another space, you should create a service key. You can do this by using the Cloud Foundry environment command line interface. For more information, see: https://docs.cloudfoundry.org/devguide/services/managing-services.html

Console command example:

Sample Code

```
cf create-service-key my-credstore my-servicekey
```

Alternatively, you can create a service key through the cockpit.

Procedure

1. Go to SAP BTP cockpit and open your account. See: Managing Subaccounts Using the Cockpit
2. Choose the *Credential Store* tile.
3. Navigate to the corresponding service instance.
4. Select *Service Keys* and choose *Create Service Key*.
5. Enter a name for the new service key.
6. (Optional) Define configuration parameters in JSON format for this service key.

   The parameters are the same as in **step 4** in task Bind a Service Instance [page 15].
7. Choose *Save*. The service key is created.

Next Steps

The properties are the same as the ones in task Bind a Service Instance [page 15].

Also, if you want to download a service key, choose  (*Save JSON to file*).

To delete an obsolete service key, choose  (*Delete Service Key*).

1.5.5 Create and Delete a Namespace [Feature Set A]

Prerequisites

> **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.

- You have enabled the SAP Credential Store for your subaccount on your Cloud Foundry space. To learn how, see: Initial Setup [page 8]
- You have created a SAP Credential Store instance. To learn how, see: Create a Service Instance [page 14]

Context

Follow the steps below to create one or multiple namespaces for your service instance.

Procedure

1. Go to SAP BTP cockpit and open your account. See: Managing Subaccounts Using the Cockpit
2. Choose the *Credential Store* tile.
3. Choose your service instance name.
4. From the left-side navigation menu, choose *Credential Store*.
5. In the default table are listed all the namespaces you have already created, if any. To create a new one, choose *Create Namespace*.
6. As a first step, you need to create a credential within the new namespace. Choose a credential type and then *Next*.
7. Enter the required information in the relevant fields.
8. (Optional) Enter a username and/or some metadata.
9. Choose *Create*.
10. The new namespace appears in the table.

Next Steps

If you want to delete an obsolete namespace, choose  (*Delete*).

Related Information

Create, Edit, and Delete a Credential [Feature Set A] [page 23]

1.5.6 Create, Edit, and Delete a Credential [Feature Set A]

Prerequisites

> **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.

- You have enabled the SAP Credential Store for your subaccount on your Cloud Foundry space. To learn how, see: Initial Setup [page 8]
- You have created a SAP Credential Store instance. To learn how, see: Create a Service Instance [page 14]
- You have created at least one namespace. To learn how, see: Create and Delete a Namespace [Feature Set A] [page 21]

Context

Follow the steps below to create one or multiple credentials for a particular namespace.

Procedure

1. Go to SAP BTP cockpit and open your account. See: Managing Subaccounts Using the Cockpit
2. Choose the *Credential Store* tile.
3. Choose your service instance name.
4. From the left-side navigation menu, choose *Credential Store*.
5. Choose a namespace from the table.
6. To create a new credential for this namespace (different from the initial one), choose *Create Credential*, and select *Password* or *Key*.
7. Enter the required information in the relevant fields.
8. (Optional) Enter a userName, credential format and/or some metadata.
9. Choose *Create*.
10. The new credential appears in the table, in the relevant tab – *Passwords (n)* or *Keys (n)*.

Next Steps

If you want to edit an existing credential, choose (*Edit*).

If you want to delete an obsolete credential, choose  (*Delete*).

Related Information

Create and Delete a Namespace [Feature Set A] [page 21]

1.5.7 Set Filters for Credentials and Namespaces [Feature Set A]

Prerequisites

> **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.

- You have enabled the SAP Credential Store for your subaccount on your Cloud Foundry space. To learn how, see: Initial Setup [page 8]
- You have created a SAP Credential Store instance. To learn how, see: Create a Service Instance [page 14]
- You have created at least one namespace. To learn how, see: Create and Delete a Namespace [Feature Set A] [page 21]

Context

If you have created more than 10 namespaces, or more than 100 credentials of a single type, they will not all be displayed in the relevant table. Thus, you can search for a particular one by using filters. The allowed search characters are letters, digits, underscore (_), dash (-), and colon (:). The supported match types are:

- Contains: **<search-string>**
- Begins with: **<search-string>***
- Ends with: ***<search-string>**
- Equals: **"<search-string>"**
- Does not equal: **!<search-string>**

> **i Note**
>
> If the search string contains invalid characters, the *Go* button will be disabled.

Procedure

1. Go to SAP BTP cockpit and open your account. See: Managing Subaccounts Using the Cockpit
2. Choose the *Credential Store* tile.
3. Choose your service instance name.
4. From the left-side navigation menu, choose *Credential Store*.
5. In the default table are listed all the namespaces you have already created.

   To filter for a particular namespace, set the filter criteria in *Search* area, and choose *Go*.
6. If you go on *Credential* level, you can set filters for passwords and keys too.

Related Information

Create and Delete a Namespace [Feature Set A] [page 21]
Create, Edit, and Delete a Credential [Feature Set A] [page 23]

1.6 REST API

Applications use the SAP Credential Store service via a REST API to list, create, read, update and delete credentials.

The REST API authenticates the applications via basic authentication. For more information, see Credential Store: REST API

Related Information

Authentication [page 25]
Encrypting Payloads [page 29]

1.6.1 Authentication

The calls to the REST API for applications are authenticated using basic authentication.

In basic HTTP authentication, a request contains a header field of the form:

```
Authorization: Basic <credentials>
```

The **<credentials>** is the base64 encoding of username and password joined by a colon.

The SAP Credential Store uses namespaces to logically isolate data stored in a service instance. The API calls are always executed in the context of a namespace. The namespace is specified with HTTP header *sapcp-credstore-namespace*.

All payloads are encrypted (or have to be encrypted) when basic authentication is used. The encryption is based on JWE compact serialization format. For more information, see Encrypting Payloads [page 29]

HTTP Request

Example

```
GET /api/v1/credentials/password?name=pass1 HTTP/1.1
Host: credstore.cfapps.sap.hana.ondemand.com
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
sapcp-credstore-namespace: namespace1
Cache-Control: no-cache
```

HTTP Respond

Example

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Length: 665
Content-Type: application/jose;charset=UTF-8
Etag: "3fdfb2a6-0d80-4142-84de-2681e1db32f6"
eyJhbGciOiJSU0EtT0FFUC0yNTYiLCJlbmMiOiJBMjU2R0NNIiwiaWF0IjoxNTQ5ODk4MTAxfQ.IaA
r_Lcv91H-
aZovfyNXvZz8IKjINqUgECdSchnReUzFYUMMw7ibd-2o5Ehc8emyv6wEFBdMBIbuQrU8a-
IHSyK_A4XYPiyTvCGUUlMvBFZaHuH1sEbBAw92C_8RUPwzlqMmb-SsKPyPu-
hVdqPcXzWV_XgCnzU87orJdZ-Fa_gH8hHC-2CdkY2fRiCRWlcnj_5JKan1yopVSQVevyLag-
JK2sV735ReMooauxZxUbxFnku0p9RHvbAB6Nv7mvu3m5rg2Z9eiuuFTB5rwMwGb2xk4XotGS5JBB1c
Puq0ZiYbPPhDDJomBG1aIGaj89PUvpwSJE1Ul5aipCFJqb3s9g.SGx8AW1GyVxoQtS8.Di3ZQJnec5
ASOoQEY4qnuOvRXVbeo1Kyp7OrKh1igBRjLky9A1dfkA-
RyECYQ1d9okcsJlcme5RUq8V21Zfxj5LMgV3MWBuP0NI_bDWQGN8v-
s5_EKXEiJmqpxW_JUzZxBrwm20CExN91DRvjKYTdL5rWIvw2GEVtEqeX3YC-
Zubavp6PWUJT7V5OIiRizLt4hJGJbTBgdglX2w.sB2qYFM6C5_uJcLfyKYDIg
```

Related Information

Encrypting Payloads [page 29]

1.6.2 Reading and Managing Credentials

Related Information

Code Example (NodeJS) [page 27]

1.6.2.1 Code Example (NodeJS)

NodeJS

Use the JavaScript code below to read, write or delete passwords and keys from service instances. The authentication to the REST API is implemented via basic credentials, and the payload encryption is mandatory enabled.

The example is using node-fetch and node-jose libraries.

Example

```
const fetch = require('node-fetch');
const jose = require('node-jose');
function checkStatus(response) {
    if (!response.ok) throw Error("Unexpected status code: " +
response.status);
    return response;
}
async function decryptPayload(privateKey, payload) {
    const key = await jose.JWK.asKey(
        `-----BEGIN PRIVATE KEY-----${privateKey}-----END PRIVATE KEY-----`,
        "pem",
        {alg: "RSA-OAEP-256", enc: "A256GCM"}
    );
    const decrypt = await jose.JWE.createDecrypt(key).decrypt(payload);
    const result = decrypt.plaintext.toString();
    return result;
}
async function encryptPayload(publicKey, payload) {
    const key = await jose.JWK.asKey(
        `-----BEGIN PUBLIC KEY-----${publicKey}-----END PUBLIC KEY-----`,
        "pem",
        {alg: "RSA-OAEP-256"}
    );
    const options = {
        contentAlg: "A256GCM",
        compact: true,
        fields: {"iat": Math.round(new Date().getTime() / 1000)}
    };
```

```
    const result = await jose.JWE.createEncrypt(options,
key).update(Buffer.from(payload, "utf8")).final();
    return result;
}
function headers(binding, namespace, init) {
    const result = new fetch.Headers(init);
    result.set("Authorization", `Basic ${Buffer.from(`${binding.username}:$
{binding.password}`).toString("base64")}`);
    result.set("sapcp-credstore-namespace", namespace);
    return result;
}
async function fetchAndDecrypt(privateKey, url, method, headers, body) {
    const result = await fetch(url, {method, headers, body})
        .then(checkStatus)
        .then(response => response.text())
        .then(payload => decryptPayload(privateKey, payload))
        .then(JSON.parse);
    return result;
}
async function readCredential(binding, namespace, type, name) {
    return fetchAndDecrypt(
        binding.encryption.client_private_key,
        `${binding.url}/${type}?name=${encodeURIComponent(name)}`,
        "get",
        headers(binding, namespace)
    );
}
async function writeCredential(binding, namespace, type, credential) {
    return fetchAndDecrypt(
        binding.encryption.client_private_key,
        `${binding.url}/${type}`,
        "post",
        headers(binding, namespace, {"Content-Type": "application/jose"}),
        await encryptPayload(binding.encryption.server_public_key,
JSON.stringify(credential))
    );
}
async function deleteCredential(binding, namespace, type, name) {
    await fetch(
        `${binding.url}/${type}?name=${encodeURIComponent(name)}`,
        {
            method: "delete",
            headers: headers(binding, namespace)
        }
    ).then(checkStatus);
}
const binding =
JSON.parse(process.env.VCAP_SERVICES).credstore[0].credentials;
let password = {
    name: "password1",
    value: "secret1",
    username: "user1",
    metadata: "{\"url\": \"https://www.example.com/path\"}"
};
console.log(await writeCredential(binding, "namespace1", "password",
password));
console.log(await readCredential(binding, "namespace1", "password",
password.name));
await deleteCredential(binding, "namespace1", "password", password.name);
let key = {
    name: "key1",
    value: jose.util.randomBytes(32).toString("base64"),
    format: "aes",
    metadata: "256 bits"
};
console.log(await writeCredential(binding, "namespace1", "key", key));
console.log(await readCredential(binding, "namespace1", "key", key.name));
await deleteCredential(binding, "namespace1", "key", key.name);
```

1.6.3 Encrypting Payloads

The SAP Credential Store provides payload encryption (enabled by default), based on JWE compact serialization format. For more information, see: JSON Web Encryption

Supported algorithms:

- Key encryption: *RSA-OAEP-256*
- Content encryption: *A256GCM*

RSA key size:

- Minimum: 2048 bits
- Maximum: 8192 bits

If you try to send an encrypted payload using different algorithms, an error will occur.

Procedure

1. For each service binding, two encryption key pairs are required. The SAP Credential Store can generate them during binding creation. The corresponding *client private key* and *server public key* will be returned in the binding properties (VCAP_SERVICES environment variable). For example:

   Example

   ```
   {
     "url": "https://credstore.cfapps.sap.hana.ondemand.com/api/v1/
   credentials",
     "username": "a17c34e3-2718-4d99-83c7-021e5b53a11d",
     "password": "mCjH5fk...",
     "encryption": {
       "client_private_key": "MIIEvAIBADANB...",
       "server_public_key": "MIIBIjANBgkqh..."
     },
     ...
   }
   ```

   Note: If the client is not an application running in SAP BTP, Cloud Foundry environment, it cannot benefit from the automatic injection of the binding properties in the environment. In this case, you can generate the client key pair locally, and then provide the public key when the service binding or the service key is created via the binding configuration parameters.

   Example

   ```
   {
     "encryption": {
       "client_public_key": "MIIBIjANBgkqh..."
     }
   ```

```
}
```

Note: For this scenario, the returned binding properties don't contain the *client_private_key* attribute.

Example

```
{
  "url": "https://credstore.cfapps.sap.hana.ondemand.com/api/v1/
credentials",
  "username": "a17c34e3-2718-4d99-83c7-021e5b53a11d",
  "password": "mCjH5fk...",
  "encryption": {
    "server_public_key": "MIIBIjANBgkqh..."
  },
  ...
}
```

2. An application should use the server public key to encrypt the payload before sending the request to the server. The following example illustrates a payload that creates or updates a password "*pass1*":

Example

```
{
  "name": "pass1",
  "value": "secret1",
  "username": "user1"
}
```

The encrypted payload, using JWE compact serialization format, will be:

```
eyJhbGciOiJSU0EtT0FFUC0yNTYiLCJlbmMiOiJBMjU2R0NNIiwiaWF0IjoxNTQ5ODg4ODkwfQ.K7U
g99QIfpmtZB_6fivr4-H0iw-8DhRL1OIJGl8K9cag-
SiMEETsHEm9_bkM6B4fq9PGWtN7SpV7XORj6tGyrqzg078KZBS9sB0zDywJ6eZYbpL3t-4dAyP6Z2a
_QqQpzaMVLZE9XRD3RUZzib03ELJ08tUDBLe0gYT4Lk4jqFtWNpAq97yUsgBTtSOM_wFGzg7uV3ewm
12yehtJPVurKpvI6MVNrH7XNYJidF-hKPbha9SGY5sB0HW1k1OqB5iPCIZk-
cP1gex7lB5IKzGi2EcHaIRzl-
JluFEk6C_Gi6N1d1wxGV5BqXrsr5reHcGHgJOjbBdEm9ltcS04IsQCHw.uc7B_dDVTe0yo9tU.H__2
OtL3SSaCR6U6KLIIeGlPnbgZzKwbgbglrTU-
c51aIw-7OERBkwTVGwF1VVBTGQCrvms.naiHDyzi8EQsbbA3IxLygw
```

The first part of the encrypted data contains JWE protected headers with the used algorithms. It also contains an additional "*iat*" header with the issue time (seconds elapsed since `01-01-1970`). An attempt to send encrypted data without the "*iat*" header or which is older than 2 minutes will result in an error. Example for *Base64* encoded JWE protected headers:

```
eyJhbGciOiJSU0EtT0FFUC0yNTYiLCJlbmMiOiJBMjU2R0NNIiwiaWF0IjoxNTQ5ODg4ODkwfQ
```

Example for *Base64* decoded JWE protected headers:

```
{"alg":"RSA-OAEP-256","enc":"A256GCM","iat":1549888890}
```

3. Finally, the request to the SAP Credential Store should contain a "*Content-Type*" HTTP header with value "**application/jose**":

Example

```
POST /api/v1/credentials/password HTTP/1.1
Host: credstore.cfapps.sap.hana.ondemand.com
Authorization: Basic ZTIwZDI...
```

```
Content-Type: application/jose
sapcp-credstore-namespace: namespace1
eyJhbGciOiJSU0EtT0FFUC0yNTYiLCJlbmMiOiJBMjU2R0NNIiwiaWF0IjoxNTQ5ODkwMjAxfQ.
juL6oZAkMWSI7cO2QRz-qO6Hqpm_FygJdUZuaqNjYuW-
Y0H9ZmhFrOdTC00nUg2rSKC9jWkpAY5CxDpdFHiveuDdHwvzIJdzOFeAaH5tJOqUoj_iD4saWXw
zgsJA_EWVmYDCLgBKuwP7Ab76_CVtN9ZNrElC-BRTxsND7rxJ0et1KHRNO0-
m463EvDfOabYJDj58iObbSIrGXjDxIGNzPkNcEZzmGhydAd5ntF5UDUgPVzCQnQ0UH2yGv_piJ1
NwGho1XKvRKvDSJ25MHQ_rqfbhf_hMaLtHlBcrqUe8f_T6KboocbL4uqXmlYPnZQ0qBED2Z1d0c
-so7YMJJ2XTCw.Mp-xqkun5d4KgwWQ.HEZSjIyMalH5bAjYWHdk1wNg3-
kle_KMEZqEWLcq3OZplg-bnvPdTfVErPna4rMrQHIA9q0.lnFLiVWFXxjsRs9t7bnWSw
```

4. The service responds with encrypted payload having the same JWE compact serialization format and the "*iat*" header.

Example

Example request:

```
GET /api/v1/credentials/password?name=pass1 HTTP/1.1
Host: credstore.cfapps.sap.hana.ondemand.com
Authorization: Basic ZTIwZDI...
Content-Type: application/jose
sapcp-credstore-namespace: namespace1
```

Example

Example response:

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Length: 665
Content-Type: application/jose;charset=UTF-8
Etag: "3fdfb2a6-0d80-4142-84de-2681e1db32f6"
eyJhbGciOiJSU0EtT0FFUC0yNTYiLCJlbmMiOiJBMjU2R0NNIiwiaWF0IjoxNTQ5ODk4MTAxfQ.
IaAr_Lcv91H-
aZovfyNXvZz8IKjINqUgECdSchnReUzFYUMMw7ibd-2o5Ehc8emyv6wEFBdMBIbuQrU8a-
IHSyK_A4XYPiyTvCGUUlMvBFZaHuH1sEbBAw92C_8RUPwzlqMmb-SsKPyPu-
hVdqPcXzWV_XgCnzU87orJdZ-Fa_gH8hHC-2CdkY2fRiCRWlcnj_5JKan1yopVSQVevyLag-
JK2sV735ReMooauxZxUbxFnku0p9RHvbAB6Nv7mvu3m5rg2Z9eiuuFTB5rwMwGb2xk4XotGS5JB
B1cPuq0ZiYbPPhDDJomBG1aIGaj89PUvpwSJE1Ul5aipCFJqb3s9g.SGx8AW1GyVxoQtS8.Di3Z
QJnec5ASOoQEY4qnuOvRXVbeo1Kyp7OrKh1igBRjLky9A1dfkA-
RyECYQ1d9okcsJlcme5RUq8V21Zfxj5LMgV3MWBuP0NI_bDWQGN8v-
s5_EKXEiJmqpxW_JUzZxBrwm20CExN91DRvjKYTdL5rWIvw2GEVtEqeX3YC-
Zubavp6PWUJT7V5OIiRizLt4hJGJbTBgdglX2w.sB2qYFM6C5_uJcLfyKYDIg
```

Related Information

Code Samples (Java) [page 32]
Code Samples (GoLang) [page 33]
Code Samples (NodeJS) [page 35]

1.6.3.1 Code Samples (Java)

Java

The Java samples below use the *jose4j* library. For more information, see Bitbucket wiki: jose4j .

> **! Restriction**
>
> Make sure you have JDK version *9* or higher. Lower Java versions don't support encryption algorithm AES256 by default. In case of lower version, you should either configure your crypto policy, or use AES128.

Encrypt

> **Example**

```
package jwe;
import java.security.KeyFactory;
import java.security.interfaces.RSAPublicKey;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import org.jose4j.jwe.ContentEncryptionAlgorithmIdentifiers;
import org.jose4j.jwe.JsonWebEncryption;
import org.jose4j.jwe.KeyManagementAlgorithmIdentifiers;
public class Encrypt {

  private static String PUBLIC_KEY = "MIIBIjAN...";
  public static void main(String[] args) throws Exception {
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    byte[] binaryKey = Base64.getDecoder().decode(PUBLIC_KEY);
    RSAPublicKey publicKey = (RSAPublicKey)keyFactory.generatePublic(new
X509EncodedKeySpec(binaryKey));
    JsonWebEncryption jwe = new JsonWebEncryption();
    jwe.setPayload("{\"name\":\"pass1\",\"value\":\"secret1\",\"username\":
\"user1\"}");

jwe.setAlgorithmHeaderValue(KeyManagementAlgorithmIdentifiers.RSA_OAEP_256);

jwe.setEncryptionMethodHeaderParameter(ContentEncryptionAlgorithmIdentifiers.A
ES_256_GCM);
    jwe.setKey(publicKey);
    long iat = TimeUnit.MILLISECONDS.toSeconds(System.currentTimeMillis());
    jwe.getHeaders().setObjectHeaderValue("iat", iat);
    String jweCompact = jwe.getCompactSerialization();
    System.out.println(jweCompact);
  }
}
```

Decrypt

Example

```
package jwe;
import java.security.KeyFactory;
import java.security.interfaces.RSAPrivateKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Base64;
import org.jose4j.jwa.AlgorithmConstraints;
import org.jose4j.jwa.AlgorithmConstraints.ConstraintType;
import org.jose4j.jwe.ContentEncryptionAlgorithmIdentifiers;
import org.jose4j.jwe.JsonWebEncryption;
import org.jose4j.jwe.KeyManagementAlgorithmIdentifiers;
public class Decrypt {
  private static final AlgorithmConstraints
CONTENT_ENCRYPTION_ALGORITHM_CONSTRAINTS = new
AlgorithmConstraints(ConstraintType.WHITELIST,
ContentEncryptionAlgorithmIdentifiers.AES_256_GCM);
  private static final AlgorithmConstraints
KEY_ENCRYPTION_ALGORITHM_CONSTRAINTS = new
AlgorithmConstraints(ConstraintType.WHITELIST,
KeyManagementAlgorithmIdentifiers.RSA_OAEP_256);

  private static final String PRIVATE_KEY = "MIIEvgIB...";

  public static void main(String[] args) throws Exception {
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    byte[] binaryKey = Base64.getDecoder().decode(PRIVATE_KEY);
    RSAPrivateKey privateKey = (RSAPrivateKey)keyFactory.generatePrivate(new
PKCS8EncodedKeySpec(binaryKey));
    JsonWebEncryption jwe = new JsonWebEncryption();
    jwe.setAlgorithmConstraints(KEY_ENCRYPTION_ALGORITHM_CONSTRAINTS);

jwe.setContentEncryptionAlgorithmConstraints(CONTENT_ENCRYPTION_ALGORITHM_CONS
TRAINTS);
    jwe.setKey(privateKey);
    jwe.setCompactSerialization("eyJhbGciOiJ...");
    Long iat = jwe.getHeaders().getLongHeaderValue("iat");
    System.out.println("iat:" + iat);
    String payload = jwe.getPayload();
    System.out.println(payload);
  }
}
```

Related Information

Encrypting Payloads [page 29]

1.6.3.2 Code Samples (GoLang)

GoLang

The GoLang samples below are using *go-jose* library. For more information, see GitHub: go-jose

Encrypt

```
package main
import (
    "bytes"
    "crypto/x509"
    "encoding/base64"
    "fmt"
    . "gopkg.in/square/go-jose.v2"
    "io/ioutil"
    "net/http"
    "time"
)
const (
    publicKey  = "MIIBIj..."
    payload    = `{"name":"pass1","value":"secret1","username":"user1"}`
)
func main() {
    publicKeyBytes, err := base64.StdEncoding.DecodeString(publicKey)
    if err != nil {
        panic(err)
    }
    pub, err := x509.ParsePKIXPublicKey(publicKeyBytes)
    if err != nil {
        panic(err)
    }
    opts := new(EncrypterOptions)
    opts.WithHeader("iat", time.Now().Unix())
    encrypter, err := NewEncrypter(A256GCM, Recipient{Algorithm: RSA_OAEP_256,
Key: pub}, opts)
    if err != nil {
        panic(err)
    }
    jwe, err := encrypter.Encrypt([]byte(payload))
    if err != nil {
        panic(err)
    }
    jweCompact, err := jwe.CompactSerialize();
    if err != nil {
        panic(err)
    }
    fmt.Println(jweCompact)
}
```

Decrypt

```
package main
import (
    "bytes"
    "crypto/x509"
    "encoding/base64"
    "fmt"
```

```
    . "gopkg.in/square/go-jose.v2"
    "io/ioutil"
    "net/http"
    "time"
)
const (
    privateKey = "MIIEvQIBAD..."
    jwe =
"eyJhbGciOiJSU0EtT0FFUC0yNTYiLCJlbmMiOiJBMjU2R0NNIiwiaWF0IjoxNTUwMDYzMTczfQ.WRy07
IjIkJ5a2f6Fl..."
)
func main() {
    privateKeyBytes, err := base64.StdEncoding.DecodeString(privateKey)
    if err != nil {
        panic(err)
    }
    privateKey, err := x509.ParsePKCS8PrivateKey(privateKeyBytes)
    if err != nil {
        panic(err)
    }
    encryptedJwe, err := ParseEncrypted(jwe)
    if err != nil {
        panic(err)
    }
    decrypted, err := encryptedJwe.Decrypt(privateKey)
    fmt.Println("response: ", string(decrypted))
}
```

Related Information

Encrypting Payloads [page 29]

1.6.3.3 Code Samples (NodeJS)

NodeJS

The NodeJS samples below are using *node-jose* library. For more information, see GitHub: node-jose

Encrypt

Example

```
const jose = require('node-jose');
const publicKeyPrefix = '-----BEGIN PUBLIC KEY-----';
const publicKeyPostfix = '-----END PUBLIC KEY-----';
const publicKey = 'MIIBIjANB...';
```

```
let examplePayload = "{\"name\":\"pass1\",\"value\":\"secret1\",\"username\":
\"user1\"}";
(async () => {
    let encryptionKey = await
jose.JWK.asKey(publicKeyPrefix.concat(publicKey, publicKeyPostfix), "pem",
{ alg: "RSA-OAEP-256"});
    let contentAlg = "A256GCM";
    let options =
        {
            contentAlg: contentAlg,
            compact: true,
            fields:
                {
                    "iat": Math.round(new Date().getTime() / 1000)
                }
        };
    const encryptedData = await jose.JWE.createEncrypt(options,
encryptionKey).update(examplePayload).final();
    console.log(encryptedData);
})();
```

Decrypt

Example

```
const jose = require('node-jose');
const privateKeyPrefix = '-----BEGIN PRIVATE KEY-----';
const privateKeyPostfix = '-----END PRIVATE KEY-----';
const privateKey = 'MIIEvAIB...';
const exampleEncryptedPayload = "eyJhbG...";
(async () => {
    let decryptionKey = await
jose.JWK.asKey(privateKeyPrefix.concat(privateKey, privateKeyPostfix), "pem",
{ alg: "RSA-OAEP-256", enc: "A256GCM"});
    const decryptedData = await
jose.JWE.createDecrypt(decryptionKey).decrypt(exampleEncryptedPayload);
    console.log(decryptedData.plaintext.toString());

})();
```

Related Information

Encrypting Payloads [page 29]

1.7 Security

This security guide provides important information on how to enable maximum security protection for a service instance, depending on the use case.

Related Information

Audit Logs [page 37]
Principle of Least Privilege [page 37]
Payload Encryption [page 38]
Rotation of Service Binding Credentials [page 38]
Data Protection and Privacy [page 38]

1.7.1 Audit Logs

The SAP Credential Store service writes audit logs for all operations with service instances, service bindings, service keys, access tokens and credentials.

Service Instance Events

- Create service instance
- Update service instance
- Delete service instance

Service Binding and Service Key Events

- Create a service binding or a service key
- Delete a service binding or a service key

Access Token Events

- Obtain access token (login)

Credential Events (password/key)

- Create a credential
- Read a credential
- Update a credential
- Delete a credential
- Delete all credentials from a namespace

1.7.2 Principle of Least Privilege

The access to a service instance and the data stored in it is based on the *principle of least privilege*. The SAP Credential Store provides two levels of authorization:

- First level – the permissions granted to a service binding or a service key. On this level, you can restrict namespace access and credential operations.
- Second level – the scopes granted to an access token.

Permissions of Service Bindings/Service Keys

These permissions are defined via the binding configuration parameters during creation time, and cannot be changed unless a service binding or a service key is re-created.

For example, if an application works only with the credentials in namespace **namespace1**, then it is highly recommend to limit the access of this application to only this namespace.

Example

```
{
    "authorization": {
        "namespace_permissions": {
            "namespace1": [
                "create",
                "read",
                "update",
                "delete",
                "list"
            ]
        }
    }
}
```

1.7.3 Payload Encryption

The SAP Credential Store provides an option to enable payload encryption and thus ensures end-to-end confidentiality of the data transferred between the client and the server. If the functionality is enabled then all request and response payloads have to be encrypted using the JWE compact serialization format. See: JSON Web Encryption

The setting is enabled by default.

For more information, see: Encrypting Payloads [page 29]

1.7.4 Rotation of Service Binding Credentials

The recommended way to achieve rotation of the binding credentials is to re-create the bindings as part of a Continuous Integration (CI) pipeline.

For example, you can use blue-green deployment model where a new binding is created for the new "green" app.

1.7.5 Data Protection and Privacy

Personal Data

The SAP Credential Store is a technical service that is not intended to store any personal data.

> **i** Note
>
> If, for any reason, an application needs to store person-related data, this application must implement all the required functions.

Account Termination

Applications are responsible for the lifecycle of the data stored in the SAP Credential Store associated with an account.

> **i** Note
>
> The applications must delete this data upon account termination. The REST API of the SAP Credential Store provides a method to delete all credentials stored in a namespace. For more information, see **API: delete all credentials** (link)

Cross-Platform Usage

The SAP Credential Store exposes REST API that can be called from Internet. If your application runs on a platform different than SAP BTP, Cloud Foundry environment, there might be contractual limitations to use this service.

1.8 Get Support

If you have questions or encounter an issue while working with the SAP Credential Store service, you can address them, using the following media:

- SAP Support Portal (An S-user is required to log in and create an incident.)
- SAP Community: Ask a question

How to create an incident?

1. Log in to SAP Support Portal.
2. Choose *Report an Incident*. *SAP ONE Support Launchpad* opens.

3. Perform a search to check whether a similar incident has already been reported.
4. If you cannot find any relevant incidents, create your own.
5. For *Component*, enter: **`BC-CP-CF-SEC-CPG`**
6. Fill in the mandatory fields.
7. Explain your problem. We recommend including the following information in the incident:
    - Region information. See: Discovery Center: SAP Credential Store
    - Cloud Foundry global account and space name
    - The URL of the page where the incident or error occurs
    - The steps or clicks used to replicate the error
    - Screenshots, videos, or the code entered

How to ask a question in SAP Community?

1. Log in to https://answers.sap.com/index.html.
2. Choose *Ask a Question*.
3. Enter the short and full text of your question or feedback.
4. Use the search to find **SAP Credential Store** and add it as a primary tag.
5. Choose *Submit your question*.
6. A page dedicated to your feedback is created. On this page, you can check for answers from SAP developers and other users.
7. If you want to receive e-mail notifications from this page, choose *Follow*.

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 a 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.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

www.sap.com/contactsap

THE BEST RUN SAP