Operations Guide
# Content

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Understanding the Basic Concepts.</strong></td>
<td>4</td>
</tr>
<tr>
<td>1.1 Runtime in Detail.</td>
<td>4</td>
</tr>
<tr>
<td>Virtual System Landscapes.</td>
<td>6</td>
</tr>
<tr>
<td>Stable URL.</td>
<td>7</td>
</tr>
<tr>
<td>Configuring Custom Domains.</td>
<td>8</td>
</tr>
<tr>
<td>Setting-Up Custom Domains in CF.</td>
<td>9</td>
</tr>
<tr>
<td>1.2 Application URLs.</td>
<td>10</td>
</tr>
<tr>
<td><strong>User Management for Cloud Integration.</strong></td>
<td>13</td>
</tr>
<tr>
<td>2.1 Creating a User for Cloud Integration.</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Managing Users and Role Assignments.</td>
<td>13</td>
</tr>
<tr>
<td>Adding Members to an Account.</td>
<td>14</td>
</tr>
<tr>
<td>Defining Authorizations for Integration Team Members.</td>
<td>16</td>
</tr>
<tr>
<td>Defining Permissions for Senders to Process Messages on a Runtime Node.</td>
<td>18</td>
</tr>
<tr>
<td><strong>Activating Enterprise Messaging.</strong></td>
<td>21</td>
</tr>
<tr>
<td><strong>Monitoring.</strong></td>
<td>23</td>
</tr>
<tr>
<td>4.1 Monitoring Message Processing.</td>
<td>25</td>
</tr>
<tr>
<td>Message Status.</td>
<td>29</td>
</tr>
<tr>
<td>Message Processing Log.</td>
<td>30</td>
</tr>
<tr>
<td>Message Processing Log - Text View.</td>
<td>33</td>
</tr>
<tr>
<td>Setting Log Levels.</td>
<td>37</td>
</tr>
<tr>
<td>4.2 Managing Integration Content.</td>
<td>39</td>
</tr>
<tr>
<td>Artifact Statuses.</td>
<td>42</td>
</tr>
<tr>
<td>Endpoint Visualization.</td>
<td>42</td>
</tr>
<tr>
<td>4.3 Managing Security.</td>
<td>43</td>
</tr>
<tr>
<td>Managing Security Material.</td>
<td>44</td>
</tr>
<tr>
<td>Managing Keystore Entries.</td>
<td>56</td>
</tr>
<tr>
<td>Managing Certificate-to-User Mappings.</td>
<td>83</td>
</tr>
<tr>
<td>Managing Access Policies.</td>
<td>85</td>
</tr>
<tr>
<td>Managing Access Policies in the CF Environment.</td>
<td>87</td>
</tr>
<tr>
<td>Managing JDBC Material.</td>
<td>89</td>
</tr>
<tr>
<td>Managing Custom Roles in the Cloud Foundry Environment.</td>
<td>94</td>
</tr>
<tr>
<td>Performing Connectivity Tests.</td>
<td>97</td>
</tr>
<tr>
<td>4.4 Managing Stores.</td>
<td>113</td>
</tr>
<tr>
<td>Managing Data Stores.</td>
<td>113</td>
</tr>
<tr>
<td>Managing Variables.</td>
<td>115</td>
</tr>
</tbody>
</table>
1 Understanding the Basic Concepts

This section provides an overview of the concepts and terms.

1.1 Runtime in Detail

For different customers separate resources (in terms of: memory, CPU, file system) of the cloud-based integration platform are allocated – although all customers might share the same hardware. This concept is also referred to as tenant isolation.

A tenant represents the resources of the cloud-based integration platform allocated for a customer. Typically one tenant is defined for each customer connected to the platform.

At runtime, data to be exchanged between the involved customers is processed on a cluster of different virtual machines hosted in the SAP cloud, at which each virtual machine is assigned to the corresponding tenant allocated for the connected customer.

A virtual machine (VM) is a software implementation of a machine that executes a program like a physical machine.

It is always made sure that the involved virtual machines are strictly separated from each other with regard to the related customers. In addition to that, each tenant uses a separate database schema which guarantees that data of different customers is strictly separated.

The architecture guarantees that different tenants are unable to interfere and that physical resources of the cloud platform are partitioned per tenant.

The runtime environment is composed of a cluster of virtual processes, where-by the message processing tasks for a tenant are performed within a dedicated Java Virtual Machine (JVM process or VM process). The individual processes are also referred to as nodes of the cluster.

A cluster is composed of different kinds of nodes.

<table>
<thead>
<tr>
<th>Kind of Node</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tenant management node</td>
<td>Performs tenant-specific management tasks like, for example, starting tenant-specific runtime nodes or deployment of artifacts like integration flows or keystores, for example.</td>
</tr>
<tr>
<td>Kind of Node</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Runtime node</td>
<td>Processes messages for a tenant. Services required for message processing like, for example, routing or mapping, are implemented as subsystems of the node.</td>
</tr>
</tbody>
</table>

**Note**
There is the option to set up multiple tenant management nodes for a tenant in order to implement failover scenarios and thus to ensure high availability. When one management node fails, one of the additional nodes can take over the tasks.

The following figure illustrates the general structure of a cluster.

![SAP Cloud Platform Integration](image)

A cluster for a tenant (shortly referred to as tenant cluster) is composed of one (or more) tenant management nodes and one or more runtime nodes.

Tenant clusters of different customers (customers) are strictly separated from each other and are unable to interfere.

In the Operations user interface (Node Explorer), the different kinds of nodes are arranged in the following way.
1.1.1 Virtual System Landscapes

There are different virtual system landscapes.

Landscapes, Virtual Servers and IP Addresses

For the list of available landscapes and respective IP addresses, see: Regions and Hosts Available for the Neo Environment.

i Note

The IP addresses are related to outbound communication, that means: for calls from the tenant to a receiver system. They are required by the customers to configure the firewall settings (IP whitelisting) to enable their system to connect to the cloud-based integration platform. Note that the table lists ranges of supported IP addresses rather than fixed IP addresses. This has the following advantage: In case of hardware problems at the cloud-based integration platforms side, the customer can quickly switch to
another machine (with another IP address) without the need to change the configuration in the back-end system.

**Ports for Outbound Communication**

When configuring an **outbound** channel (receiver adapter), make sure to use the following standard ports:

- Outbound HTTP/HTTPS connections: By default, **port 443 and all ports > 1024** are opened in the SAP Cloud Platform firewall. In case of any issues, open a ticket (component **LOD-HCI**).
- For **SFTP** connections, make sure that the SSH data channel between the SAP cloud platform and the SFTP server is opened. Use **port 22** at SAP Cloud Platform side.
- For **SMTP** connections, use **port 25**.

### 1.1.2 Stable URL

You can use this URL to access the tenant.

**Management URL**

<table>
<thead>
<tr>
<th>Tenant</th>
<th>SSL Host</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Productive</td>
<td>HCI</td>
<td>https://&lt;Account Short Name&gt;-tmn.hci.&lt;Landscape Host&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, <a href="https://0001-tmn.hci.eu1.hana.ondemand.com">https://0001-tmn.hci.eu1.hana.ondemand.com</a></td>
</tr>
<tr>
<td>Test</td>
<td>HCI</td>
<td>https://&lt;Account Short Name&gt;-tmn.hci.&lt;Landscape Host&gt;</td>
</tr>
</tbody>
</table>

**Runtime URL**

<table>
<thead>
<tr>
<th>Tenant</th>
<th>SSL Host</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Productive</td>
<td>HCISBP</td>
<td>https://&lt;Account Short Name&gt;-ifl-map.hcisbp.&lt;Landscape Host&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, <a href="https://0001-ifl-map.hcisbp.eu1.hana.ondemand.com">https://0001-ifl-map.hcisbp.eu1.hana.ondemand.com</a></td>
</tr>
<tr>
<td>Test</td>
<td>HCISBT</td>
<td>https://&lt;Account Short Name&gt;-ifl-map.hcisbt.&lt;Landscape Host&gt;</td>
</tr>
</tbody>
</table>

**Note**
- In the URL, account short name refers to account detail mentioned in SAP email. You receive this email after onboarding the tenant. Also, you can access the following link to see the list of available landscapes and respective IP addresses: [Landscape Hosts](https://0001-ifl-map.hcisbt.eu1.hana.ondemand.com).
1.1.3 Configuring Custom Domains

Allows you to customize the default tenant URL or domain as per your needs and access the tenant using your own the domain.

There are use cases where you do not wish to expose the default domain provided by SAP Cloud Platform Integration. In such a scenario, you can construct custom domain names and TLS settings for specific Cloud Integration instance.

Use the procedure here to create and add custom domains.

Note
When a custom domain is used, the domain name and the server certificate of the domain are owned by the customer.

1. To use a custom domain for your application, you must fulfill a number of preliminary steps. For more information, see Prerequisites.
2. Create an SSL Host - the host holds the mapping between your chosen custom domain and the application on SAP Cloud Platform as well as the SSL configuration for secure communication through this custom domain.
3. Upload a Certificate - it will be used as a server certificate on the SSL host.
4. Bind the Certificate to the SSL Host.
5. Add the Custom Domain - this maps the custom domain to the Cloud Integration URL.

Note
The format of the default application URL is https://<application_name><provider_subaccount>-<consumer_subaccount>.<domain>.

You can find the required components for the above URL in the SAP Cloud Platform Cockpit:

<table>
<thead>
<tr>
<th>URL Elements</th>
<th>Where to find it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;application_name&gt;</td>
<td>In Application &gt; Subscriptions find the subscribed application details.</td>
</tr>
<tr>
<td>&lt;provider_subaccount&gt;</td>
<td>In Application &gt; Subscriptions and then choose the relevant Application. In the Subscribed Application - &lt;account ID&gt; Overview page, find the Provider Subaccount details. For example, Provider Subaccount: myacct (wc077ad).</td>
</tr>
</tbody>
</table>
URL Elements | Where to find it?
--- | ---
<consumer_subaccount> | Under the Subaccount Information section, use the subaccount Name.

**Note**

After creating an custom domain, you can use both the default URL as well as the custom domain to access the application.

6. **Configure DNS** - you can create a CNAME mapping.
7. **Configure Single Sign-On** - if you have a custom trust configuration in your subaccount, you need to enable single logout.

The configuration of custom domains has different setups related to the subscriptions of your subaccount. For more information about custom domains for applications that are part of a subscription, see [Custom Domains for Multitenant Applications](https://example.com).

### Enabling TLS Protocol Versions

Configure and enable TLS protocol of your choice for inbound communication to SAP Cloud Platform Integration. For more information, see [How to replace SSL with TLS](https://example.com).

**Note**

Use `--supported-protocols` parameter to enable the specified TLS protocol.

### 1.1.4 Setting-Up Custom Domains in CF

Allows you to customize the default tenant URL.

There are use cases where you don’t wish to expose the default domain provided by SAP Cloud Platform Integration. In such a scenario, you can construct custom domain names and TLS settings for a specific Cloud Integration instance. In the following you can configure two Custom Domains, one for accessing the web-UI and one for the runtime worker, for example, `webui.subdomain.mydomain.com` and `runtime.subdomain.mydomain.com`.

1. **Initial Steps:**
   To use the Custom Domain Service you have to complete some initial steps, described here: [Initial Setup](https://example.com).
2. **Prerequisites**
   Before configuring custom domains, you have to fulfill a number of prerequisites. See: [Prerequisites](https://example.com).
3. **Configuring Custom Domains**
   1. To get a better overview of the creation process of custom domains with TLS/SSL server authentication, see: [Creating Custom Domains with TLS/SSL Server Authentication](https://example.com).
      1. To create custom domains, see: [Create Custom Domains](https://example.com).
2. To create a certificate signing request, see: Create a Certificate Signing Request. We recommend to create a wildcard certificate for both custom domains.

3. To upload and activate the signed certificate, see: Upload and Activate the Signed Certificate.

4. To configure the DNS for a custom domain, see: Configure the DNS for a Custom Domain.

2. To map the Cloud Integration instance to a custom domain, see: Map a SaaS Application to a Custom Domain.

Mapping example for the web-UI:

```
cf custom-domain-map-route <tenantname.landscape-id.cfapps.datacentername.hana.ondemand.com> <webui.subdomain.mydomain.com>
```

Mapping example for the runtime:

```
cf custom-domain-map-route <tenantname.landscape-runtimeid.cfapps.datacentername.hana.ondemand.com> <runtime.subdomain.mydomain.com>
```

You have to execute both mappings, one for the web-UI and one for the worker node.

3. To test your custom domain, and make sure it’s reachable and secured by a certificate, open your custom domain web-UI URL in a browser. Don’t open the runtime URL in the web browser, but rather send a postman request to the runtime URL.

Remember that certain steps during the setup process of your custom domain can take up to 24 hours and the DNS configuration can take several hours depending on your provider.

4. To add trusted certificates for client authentication to a custom domain, see: Adding Trusted Certificates for Client Authentication to a Custom Domain.

4. Managing Custom Domains

To manage custom domains, check the steps described in: Managing Custom Domains.

5. Removing the Custom Domain Service Instance

To remove the Custom Domain Service Instance, see Remove the Custom Domain Service Instance.

If you encounter issues with your custom domains, we recommend that you follow the procedure described in Monitoring and Troubleshooting.

### 1.2 Application URLs

Application URLs provide various options to connect to the SAP Cloud Platform Integration virtual environment.

Application URLs are defined both for the tenant management node application and for the runtime node application.
The following table provides a summary of the available application URLs:

<table>
<thead>
<tr>
<th>Node Type</th>
<th>Application URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tenant management node</td>
<td>http://&lt;tenant management node address&gt;/Operations</td>
<td>Operations URL to be used when connecting to the tenant using Eclipse (for example, when using the Adapter Development Kit)</td>
</tr>
<tr>
<td>Tenant management node</td>
<td>http://&lt;tenant management node address&gt;/api</td>
<td>Base URL to access the SAP Cloud Platform Integration application programming interface (API) Note that the SAP Cloud Platform Integration OData API is published on the API Business Hub at: <a href="https://api.sap.com/shell/discover/content-package/CloudIntegrationAPI">https://api.sap.com/shell/discover/content-package/CloudIntegrationAPI</a></td>
</tr>
<tr>
<td>Tenant management node</td>
<td>http://&lt;tenant management node address&gt;/itspaces</td>
<td>Web UI URL for the tenant (for dialog users to design integration flows, monitor message processing, and execute additional tasks within an integration project)</td>
</tr>
<tr>
<td>Tenant management node</td>
<td>http://&lt;tenant management node address&gt;/cxf</td>
<td>Endpoint address under which SAP Solution Manager can retrieve alert notifications (if configured accordingly)</td>
</tr>
<tr>
<td>Runtime node</td>
<td>http://&lt;runtime node address&gt;/as2</td>
<td>AS2 sender adapter endpoint base URL</td>
</tr>
<tr>
<td>Runtime node</td>
<td>http://&lt;runtime node address&gt;/cxf</td>
<td>SOAP sender adapter endpoint base URL</td>
</tr>
<tr>
<td>Runtime node</td>
<td>http://&lt;runtime node address&gt;/gw/odata</td>
<td>OData endpoint URL</td>
</tr>
<tr>
<td>Runtime node</td>
<td>http://&lt;runtime node address&gt;/http</td>
<td>HTTPS sender adapter endpoint base URL For example, assume that you have configured an integration flow with an HTTPS sender adapter and as Address you have specified the following value: /myEndpoint. Then an HTTP client can call this integration flow through the following endpoint address: http://&lt;tenant address&gt;/http/myEndpoint</td>
</tr>
</tbody>
</table>
To display the available application URLs, open SAP Cloud Platform Cockpit, select Subscriptions and then the required application (either the tenant management node application ending with tmn or the runtime node application ending with iflmap).
2 User Management for Cloud Integration

Users management tasks are required for different activities associated with the setup and operation of the cluster.

- To create users, you register at blogs.sap.com.
- Managing user-to-role assignments (using SAP Cloud Platform Cockpit). This task includes initial activities by the tenant administrator to define authorizations for people who are supposed to work on the tenant cluster.

Related Information

Creating a User for Cloud Integration [page 13]
Managing Users and Role Assignments [page 13]

2.1 Creating a User for Cloud Integration

For several tasks associated with SAP Cloud Platform Integration, you need a user account on blogs.sap.com.

Context

There are two types of users who can access SAP Cloud Platform Integration:

1. **S-User**: If you have an S-User ID, you are automatically a member of blogs.sap.com.
2. **P-User**: If you do not have an S-User ID, you can create a public user (P-User) account directly by referring to this blog.

After you have created a new user account, you can find your user ID in your account settings.

2.2 Managing Users and Role Assignments

You specify the members of the account and assign roles to them.

If this function is not available for your account, ask your SAP contact or create a ticket (component LOD-HCI) to activate it.
This task includes initial activities by the tenant administrator to assign roles to the users associated with all people who are supposed to work on the tenant cluster.

The task is subdivided into the following main activities:

- **Adding members to the account:**
  With this step, you specify all users who should have assigned the same role like the tenant administrator. In addition to that, you can also define all users who should have a restricted tenant administrator role (on the account). This specific role allows you to assign application-specific roles to other users of the account (like, for example, the integration developer role).

- **Defining authorizations for individual integration team members:**
  With this step, you assign the actual application-specific roles to the integration team members (like, for example, the integration developer role).

To perform these steps, open the SAP Cloud Platform Cockpit using the S-user ID provided to you by SAP (in the mail that contains also the information on the account).

The URL of the SAP Cloud Platform Cockpit depends on the data center.

### SAP Cloud Platform Cockpit URLs

<table>
<thead>
<tr>
<th>Region</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europe (Rot)</td>
<td><a href="https://account.hana.ondemand.com/cockpit">https://account.hana.ondemand.com/cockpit</a></td>
</tr>
<tr>
<td>US East (Ashburn)</td>
<td><a href="https://account.us1.hana.ondemand.com/cockpit">https://account.us1.hana.ondemand.com/cockpit</a></td>
</tr>
<tr>
<td>Australia (Sydney)</td>
<td><a href="https://account.ap1.hana.ondemand.com/cockpit">https://account.ap1.hana.ondemand.com/cockpit</a></td>
</tr>
</tbody>
</table>

### Related Information

- Adding Members to an Account [page 14]
- Defining Authorizations for Integration Team Members [page 16]

### 2.2.1 Adding Members to an Account

You specify the members of the account to define who is involved in an integration project.

### Context

Perform the following steps.
Procedure

1. To specify all users who should get assigned the tenant administrator role, open the SAP Cloud Platform Cockpit using the S-user ID provided to you by SAP (in the mail that contains also the information on the account).
2. Choose Members ➤ Add Members.
3. Enter the user ID.
4. Select the role which should be assigned to the user.

Assign role Administrator to the user who is supposed to have the full permissions of an administrator.
Assign role Application User Admin to the user who is supposed to have restricted administrator permissions.
These roles are published by SAP Cloud Platform - they are not specific to any tasks related to integration projects. When you define more specific authorizations for integration team members, you assign other roles (as described in the corresponding section).

5. Choose Add Members.

Related Information

Defining Authorizations for Integration Team Members [page 16]
Support Tasks [page 190]

2.2.2 Defining Authorizations for Integration Team Members

To authorize selected people to work on the account as part of the integration team in the context of SAP Cloud Platform Integration (for example, as integration developers), you assign roles to the associated users.

Context

→ Remember
There are currently certain limitations when working in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.

Authorizations can also be given to users that are not associated with any integration team member (for example a user associated with a system that is to be authorized to send a message to Cloud Integration).

Perform the following steps for all users for which to assign authorizations.

i Note
You can also define Role Collections and assign roles to role collections. That way, you can assign roles at once for all users who should to get identical permissions on the account.

i Note
SAP has provided you with the link to access SAP Cloud Platform Cockpit.
Procedure

1. In SAP Cloud Platform Cockpit, select your subaccount and then choose Security ➤ Authorizations. Enter your user ID, and choose Assign.

2. In the navigation pane, choose Security ➤ Authorizations.

3. If you like to first create a user group, on page Authorization Management choose the tab Groups. Enter a group name and add the relevant users to it (as shown in the figure).

4. On page Authorization Management enter the user group or user for which you like to assign authorization groups.

5. Choose Assign.

6. On page Assign Roles to User <User Name> choose the authorization group to be assigned.

Note: Changes will affect new sessions only.
You can assign groups of roles (also referred to as **authorization groups** and starting with `AuthGroup`) or individual roles. Authorization groups cover the different tasks associated with an integration project.

As *Application*, choose the one which ends with `tmn` (for tenant management node).

7. Choose **Save**.

### 2.2.3 Defining Permissions for Senders to Process Messages on a Runtime Node

**Context**

→ **Remember**

There are currently certain limitations when working in the Cloud Foundry environment. For more information on the limitations, see SAP Note [2752867](#).

↓ **Note**

SAP has provided you with the link to access SAP Cloud Platform Cockpit.

↓ **Note**

In order to authorize a sender system to call a tenant (using HTTPS/basic authentication) and to get messages processed on the tenant, you need to assign to the associated technical user the specific role `ESBmessaging.send`.

**Procedure**

1. In SAP Cloud Platform Cockpit, select your subaccount and then choose **Security ➤ Authorizations**
   Enter your user ID, and choose **Assign**.

2. In the navigation pane, choose **Security ➤ Authorizations**

3. On page **Authorization Management**, enter the user for which you like to define the permission.
4. Choose **Assign**.

5. As **Application**, choose the one, which ends with `iflmap` (corresponding to a runtime node of the cluster that actually is in charge of processing the message).

6. Select the role **ESBmessaging.send**.

   **Assign roles to user** myUser

   **Account:** avrhc1
   **Application:** iflmap
   **Role:** ESBMessaging send

   *Note: Changes will affect new sessions only.*
Next Steps

**i Note**

When configuring inbound authorization settings in the corresponding sender adapter, you can keep the role name `ESBmessaging.send` pre-entered by default in the User Role field (when you have for Authorization selected the option User Role).

Alternatively, you can define a custom role for the runtime node by performing the following steps:

1. Select your subaccount.
2. Choose Subscriptions.
3. Under Application, select the one ending with `iflmap` (for your runtime node).
4. Choose Roles.
5. Choose New Role and enter the name of your custom role.

**i Note**

When creating a new user role, be aware of the fact that for role names in the Cloud Foundry environment there are certain restrictions. In the Cloud Foundry environment, only the following characters are allowed in user role names:

- a-z
- A-Z
- 0-9
- _ (underscore)
- . (period)

If you don't consider these restrictions when creating a new role in the Neo environment (as described here), you run into problems when migrating your integration content to the Cloud Foundry environment.

**i Note**

When configuring inbound authorization settings in the corresponding sender adapter, you have to enter this role name in the User Role field (when you have for Authorization selected the option User Role).
3 Activating Enterprise Messaging

You can activate Enterprise Messaging on all licensed versions of Cloud Integration.

Prerequisites

- You must have purchased a minimum of 10 SKUs of SAP Enterprise Messaging, in which five SKUs are consumed by Message Monitor and remaining five SKUs are used to activate JMS capabilities.
- Contact your Account Executive for more details and for purchasing additional SKUs for SAP Enterprise Messaging.

Context

Use procedure to activate Enterprise Messaging for consuming JMS capabilities.

Procedure

1. The following steps are executed by Global Account Administrator:
   a. Log into the accounts cockpit (SAP Cloud Platform Cockpit).
   b. Choose a global account.
   c. Navigate to Accounts tab.
   d. Choose a customer account.
   e. Navigate to Entitlements page.
   f. Choose Edit to assign the required quota to a relevant subaccount in the Enterprise Messaging section.
   i Note
   - The quota must be in 1:5 ratio, where one messaging queue on Cloud Integration corresponds to five quotas.
   g. Saves the changes before modifying the quota for other subaccounts.

2. Execute following steps to assign tenant administrator role:
   a. Navigate to Subscription tab.
   b. Choose Provision application for hcisvc provider account.
   c. Navigate to Roles tab.
   d. Assign AuthGroup.Administrator authgroup to activate the application.

3. The following steps are executed by Tenant Administrator to activate enterprise messaging and to modify the messaging queues:
a. Navigate account cockpit.
b. Navigate to Services tab.
c. Choose Process Integration tile.
d. Choose Configure Process Integration link under Take Action.
e. Choose Enterprise Messaging tab, and then choose Activate to start the service.

**Note**
- Google Chrome, Firefox and Safari web browsers support provisioning application.
- Choose Deactivate to deactivate the messaging service.

Perform the steps here to increase the enterprise messaging queues for a specific subaccount.

a. Communicate with the Global Account Administrator to increase the quota to the relevant subaccount.
b. In the Cloud Platform Integration application, select Enterprise Messaging tab.
c. Choose Edit to increase the message queues.

Perform the steps here to decrease the enterprise messaging queues for a specific subaccount.

a. Decrease the message queues by selecting Enterprise Messaging and choose Edit to decrease the message queues.

**Note**
- You must remove unused JMS queues from your tenant before decreasing the message queues.
- For more information, see Managing Message Queues [page 116].

b. Communicate with the Global Account Administrator to decrease the quota to the relevant subaccount in the Entitlements page of the account cockpit.

For more information, read the blog on Activating and Managing Enterprise Messaging Capabilities ➤.
An integration developer can use a Web user interface to check the status of messages and integration content artifacts for a tenant cluster.

The start page is subdivided into the following sections, each covering a specific task area.

### Sections on Monitoring Start Page

<table>
<thead>
<tr>
<th>Container</th>
<th>Allows you to ...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Monitor Message Processing</strong></td>
<td>Monitor message processing on the tenant.</td>
</tr>
<tr>
<td></td>
<td>Tiles in this section show the number and status of processed messages within a specified time window.</td>
</tr>
<tr>
<td><strong>Manage Integration Content</strong></td>
<td>Manage integration content for the tenant.</td>
</tr>
<tr>
<td></td>
<td>Tiles in this section show the number and status of integration content artifacts (such as integration flows).</td>
</tr>
<tr>
<td><strong>Manage Security</strong></td>
<td>Manage security artifacts for the tenant.</td>
</tr>
<tr>
<td></td>
<td>Tiles in this section allow you to manage certain tasks related to the setup of secure connections between your tenant and remote systems.</td>
</tr>
<tr>
<td></td>
<td>The Security Material tile provides access to and allows you to deploy security-related artifacts such like User Credentials artifacts.</td>
</tr>
<tr>
<td></td>
<td>The Keystore tile provides access to the content of the tenant keystore and allows you to manage its content and also the lifecycle of keys and certificates.</td>
</tr>
<tr>
<td></td>
<td>The Certificate-to-User Mappings tile allows you to manage certificate-to-user mappings (relevant for the setup of inbound connections).</td>
</tr>
<tr>
<td></td>
<td>The Connectivity Tests tile allows you to test the connectivity to a receiver system.</td>
</tr>
</tbody>
</table>
Container | Allows you to ...
--- | ---
Manage Stores | Manage temporary data storages on the tenant.

The **Data Stores** tile provides an overview of storages on the tenant, which are temporarily used to persist data of different kind during message processing (when using the Data Store Operations step type).

The **Variables** tile allows you to monitor variables used in integration flows.

The **Message Queues** tile allows you to monitor queues that are active for a tenant.

**i Note**
You can only monitor message queues if a specific broker has been provisioned.

The **Number Ranges** tile provides an overview of number ranges that are used in business-to-business scenarios.

| Access Logs | Tiles in this section allow you to monitor audit logs (resulting from system changes) and to analyze errors that occurred during inbound HTTP processing (and documented in system log files).

| Manage Locks | This section allows you to display and manage lock entries that are created (in the in-progress repository) to avoid the same message being processed several times in parallel.

Each section contains tiles, which show filter settings and the number of messages/artifacts that correspond to the filter settings.

Clicking a tile opens a page with more information.

The start page is automatically refreshed every 5 seconds. The autorefresh ends after 5 minutes (indicated by a message).

**i Note**
All times displayed on the Monitoring pages are local times (with regard to the client of the Web application).

### Managing Tiles

You can customize the start page to fit your personal needs by adding new tiles (by clicking an empty tile containing a + symbol) or deleting existing ones. You can also rearrange the start page by dragging and dropping tiles to another location. For example, you can customize the start page so that it shows only data related to integration flows that are edited by you.
You can rearrange tiles only within the same section.

You can add, delete, or edit tiles in the following section types:

- *Monitor Message Processing*
- *Manage Integration Content*

**Note**

The personal settings are persisted in the client of the user. This means that you can log out, close the browser, and log on again to Web monitoring, and the previously specified user settings are kept.

To edit a tile, right-click the tile and choose *Edit*.

When you add or edit a tile, you can specify the following filter categories:

- For *Monitor Message Processing* tiles you can specify *Status, Time*, and the related *Integration Flow* for the messages to be displayed.
  - In the dropdown list for the *Integration Flow*, the display name of the integration flow is shown. The tooltip for an entry shows the technical name and the version of the corresponding integration flow.
- For *Manage Integration Content* tiles, you can specify the *Status* and the *Type* of the integration content artifact to be displayed.

**Related Information**

- Monitoring Message Processing [page 25]
- Managing Integration Content [page 39]
- Managing Security [page 43]
- Managing Stores [page 113]
- Accessing Logs [page 124]
- Managing Locks [page 126]

### 4.1 Monitoring Message Processing

The message monitor provides an overview of the messages processed on a tenant and allows you to display the details for individual messages.

**→ Remember**

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note [2752867](https://support.sap.com/notes/2752867).

You open the message monitor by clicking a tile in the *Monitor Message Processing* area.

Messages are displayed according to the filter settings of the tile.
Filter Settings

You can control which messages are displayed by changing the filter.

You can filter messages by *Time, Status, Artifact*.

Filter Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td>Allows you to select from the following predefined time intervals:</td>
</tr>
<tr>
<td></td>
<td>● <em>All</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Past Minute</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Past Hour</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Past 24 Hours</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Past Week</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Past Month</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Custom</em></td>
</tr>
<tr>
<td></td>
<td>You can select the start and end time of the interval.</td>
</tr>
<tr>
<td></td>
<td>The specified time interval is displayed above the message list. When you browse different pages of the message monitor, the time interval stays the same. You can modify the time interval only by changing the filter settings or by refreshing the message monitor (applies to all-time intervals except for <em>Custom</em>).</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Allows you to filter messages according to their status.</td>
</tr>
<tr>
<td></td>
<td>In the dropdown list you can select one of the following values as the status:</td>
</tr>
<tr>
<td></td>
<td>● <em>All</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Failed</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Retry</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Completed</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Processing</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Escalated</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Cancelled</em></td>
</tr>
<tr>
<td></td>
<td>● <em>Discarded</em></td>
</tr>
<tr>
<td></td>
<td>For more information about the statuses, see the Related Links section.</td>
</tr>
<tr>
<td><strong>Artifact</strong></td>
<td>Allows you to display messages associated with a specific artifact.</td>
</tr>
<tr>
<td></td>
<td>The dropdown list contains all artifacts deployed on the tenant.</td>
</tr>
<tr>
<td></td>
<td>You can filter for artifacts with a specific sequence of characters in their name or ID. The search is case-insensitive.</td>
</tr>
<tr>
<td></td>
<td>The tooltip shows the technical name and the version of the artifact.</td>
</tr>
</tbody>
</table>

Go to *Use More Fields* to display addition filter criteria, such as:

- *Sender*
- *Receiver*
- Custom Status
- Application Message Type

Alternatively you can search for IDs in the ID field, and display messages associated with various IDs such as:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message ID</td>
<td>Identifies the message uniquely.</td>
</tr>
<tr>
<td>Correlation ID</td>
<td>Identifies correlated messages</td>
</tr>
<tr>
<td>Application ID</td>
<td>Is set when an SAP_ApplcationID header element is specified in the associated integration flow in the Content Modifier step.</td>
</tr>
</tbody>
</table>

If you search by ID, the system checks whether there's a message with the specified Correlation ID. If no such message is found, the system searches for messages with the given Application ID. This search attribute can't find messages where the Correlation ID is the same as an existing Message ID, or the Application ID equals a Correlation ID or Message ID (by coincidence).

Searching messages by this attribute is helpful for support use cases (to do root cause analyzes, for example).

**Messages Table**

The messages for the selected filter settings are displayed in a table (under Messages). If the number of filtered messages exceeds 50, the list is split over several pages (each page containing a maximum of 50 messages).

You can browse through the different pages by selecting the corresponding navigation options (Show first data page, Show last data page, Show previous data page, and Show next data page).

Alternatively, you can enter a page number to navigate to a specific page of the list.

The following attributes are displayed for each message:

<table>
<thead>
<tr>
<th>Attribute in Message Overview</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artifact Name</td>
<td>Display name of the artifact (for example, the name of the integration flow that specifies the message processing). The tool-tip shows the technical name, the version of the artifact, and the package name.</td>
</tr>
<tr>
<td>Status</td>
<td>Status of end-to-end message processing.</td>
</tr>
<tr>
<td>Last Updated at</td>
<td>Time at which the message processing log was last updated.</td>
</tr>
<tr>
<td>Processing Time</td>
<td>Total message processing time.</td>
</tr>
</tbody>
</table>

For a selected message, the details are displayed to the right of the message table. The header area provides the following information about the selected message: Artifact Name and Last Updated at.
Below the header, the following sections contain detailed information about the selected message:

- **Status**
  Contains status information on the message such as the Processing Time. If the message is **NOT** in status COMPLETED, the last error message is displayed (if available).

- **Properties**
  Contains
  - the **Message ID**
  - the **Correlation ID**
  - the **Sender**
  - the **Receiver**
  - the **Application Message ID**
  - the **Application Message Type**
  - the **Custom Status**

  **i Note**
  *Sender, Receiver, Application Message ID, and Application Message Type* and the **Custom Status** are optional and can be set via the Content Modifier. They’re only visible if they aren’t empty.
  The **Custom Status** is visible only in case it differs from the overall processing status.

- **Custom Header Properties**
  The properties can be set via script-step. If they’re set, they’re displayed in this section.
  For more information, see:

- **Logs**
  Displays the **Log Level** and the **Runtime Node**. Clicking the log level link takes you to the message processing log displayed in table form. You access the textual representation by selecting **Open Text View**.
  If the processing of an integration flow failed and retry runs were performed, the runs are displayed in a table (up to 50 retries) in the **Logs** section.

- **Attachments**
  This section is only displayed if the message processing log contains attachments.
  Any type of attachment that can be displayed as text is supported. In particular, the following types of content can be displayed:
  - Plain text
  - XML
    XML attachments are displayed as formatted plain text (including tags).
  - Text files with comma-separated values
  - Text files with tab-separated values
    In these files, each entry is represented as one line of the text file.
  - HTML
    HTML attachments are displayed as plain text (including tags).

- **Artifact Details**
  This section displays the artifact **Name**, **ID**, **Type**, and the **Package** if available. You can open the integration flow by clicking the artifact name link.
4.1.1 Message Status

Message Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPLETED</td>
<td>Message has been delivered to receiver successfully.</td>
</tr>
<tr>
<td>PROCESSING</td>
<td>Message is currently being processed.</td>
</tr>
<tr>
<td>RETRY</td>
<td>Status retry is set if an error occurred during message processing, and a retry was automatically started.</td>
</tr>
<tr>
<td>ESCALATED</td>
<td>During message processing an error occurred, and no retry has been triggered. For synchronous messages, an error message is sent to the sender.</td>
</tr>
<tr>
<td>FAILED</td>
<td>Message processing failed, message hasn’t been delivered to receiver, and no retries are possible. In other words: FAILED is a final status, and message processing ultimately has failed.</td>
</tr>
<tr>
<td>CANCELLED</td>
<td>Manual cancellation of entries in the JMS queue - MPL is set to status cancelled.</td>
</tr>
</tbody>
</table>

Note

In general, the last final status is accepted, but if the message processing is still ongoing, and the status CANCELLED is set, it’s overwritten by another “final status” such as COMPLETED, after completion of the message processing.

DISCARDED

For scheduler triggered integration flows, the MPL is shown on the worker node where the message processing started first. For all subsequent message processing starts, the message status is set to DISCARDED.

An aggregated message processing log (MPL) can have the following status values:

Status of Aggregated Messages

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCESSING</td>
<td>Is set as soon as the aggregation process is started and remains as long as the aggregate is still open for further messages. Note that this status can in many situations be shown for quite some time (even days) – depending on the applied aggregation use case.</td>
</tr>
</tbody>
</table>
### 4.1.2 Message Processing Log

The message processing log displays structured information of a message.

Select a message from the list. In the view section, the header shows the integration flow name, the processing date and the time of its last update.

Below, you can select and display Status, Properties and Logs. If the processing of an integration flow failed, the retry runs performed are displayed in the Logs section.

The logs section shows the number of runs as well as the following information:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Displays the sequential number of the run</td>
</tr>
<tr>
<td>Started At</td>
<td>Displays the date and time of the last run.</td>
</tr>
<tr>
<td>Duration</td>
<td>Displays the duration of the processing run in ms.</td>
</tr>
<tr>
<td>Log Level</td>
<td>Displays the log level.</td>
</tr>
<tr>
<td>Process ID</td>
<td>Displays the ID of the worker node on which the run was executed.</td>
</tr>
</tbody>
</table>
You open the message processing log by clicking on the log level of one of the runs.

The detailed view shows the list of the Run Steps on the left.

If an error occurred in one of the run steps, it is shown by a red exclamation mark in the run step list. To display the error message, click on the exclamation mark.

If a Split or Multicast Step is used, the steps belonging to one sub-message get the same segment number. They can therefore be grouped together.

To view the Log Content, select one run step from the list and choose Log Content. The log content displays the Properties and the Activities of the step. If more than one step is related to a flow element, all of them are selected within the list and you can scroll from one step to the next by using the arrows in the element info bar and view the properties and activities accordingly.

The Message Content section is only visible, if the log level has been set to trace.

**i Note**

The names of the flow elements can only be shown in the run step list if the integration flow model is loaded. If the model cannot be loaded, the system only shows the run steps IDs.

The Integration Flow Model is displayed on the right hand side, if it is deployed.

You can view the path taken by the message in the integration flow. If you select a run step for the list, it is highlighted in the integration flow model.

You can also select an element of the Integration Flow Model and the related run step will be highlighted in the run step list on the left hand side.
The properties view of the selected integration flow element is displayed at the page bottom.
The availability of the trace data is also visualized in the integration flow model as envelope icons.

### Envelope Icons

<table>
<thead>
<tr>
<th>Icons</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="example" alt="icon" /></td>
<td>Successfully processed step with trace data available.</td>
</tr>
<tr>
<td><img src="example" alt="icon" /></td>
<td>Error in processed step with trace data available.</td>
</tr>
<tr>
<td><img src="example" alt="icon" /></td>
<td>Successfully processed step with no trace data available.</td>
</tr>
<tr>
<td><img src="example" alt="icon" /></td>
<td>Error in processed step with no trace data available.</td>
</tr>
</tbody>
</table>

In order to have a visual distinction between steps with available trace data and steps with processing information only, inverted icons are displayed if there is no trace data available.  for a successfully processed step and if the step processing resulted in an error.

**Note**
If the log level is set to *Info* and the processing does not show any error, the link to the detailed view is not active. If the processing shows at least one error, the link is active and you can access the detailed view.

**Note**
This component stores data on your tenant. Note that the tenant space is limited and shared with other tenant data like message processing logs.

### Related Information

- Message Processing Log - Text View [page 33]
- Monitoring Message Processing [page 25]
- Message Processing Log - Adapter Tracing [page 32]
- Setting Log Levels [page 37]
- Message Status [page 29]

#### 4.1.2.1 Message Processing Log - Adapter Tracing

The adapter tracing is part of the regular tracing feature and the payloads are recorded if you have set the log level to *Trace*.

The adapter tracing is only possible for adapters that transform the message either before sending or upon reception, such as *AS2, Ariba, LDAP, Mail, SuccessFactors* and *HTTP/HTTPS*. You also get tracing information
for integration flows containing CFX based adapters such as IDOC, SOAP, SAPRM, but in this case you have to redeploy the integration flow to get the tracing data recorded in the regular tracing feature and displayed.

Adapter tracing is not possible for SMS, SFTP, Facebook, Twitter and Process direct, as they do not modify the payload.

→ Remember

For log level Trace, detailed information is recorded for all steps and in addition, the message content is tracked. The trace function expires after a certain time (default value: 10 minutes). After expiry the log level switches back to the log level set before. The recorded message content is also retained for a certain time (default value: 1 hour).

To show the trace information details, select an integration flow from the overview list, go to the Log section and open the link by clicking the log level.

In the integration flow model, the trace entries are displayed as little filled envelopes.

If the payload was recorded for a specific step of an adapter, you can open it the Message Content of the details page of the processing step. The payload is displayed with header and content. You can also download it by clicking on the Download button on top right corner. This provides a zip file containing one header and one payload file.

### 4.1.3 Message Processing Log - Text View

The message processing log displays structured information on the processing of a message.

→ Remember

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.

The message processing log comprises a log header and log steps. In the log header you find more general information. The log steps show specific processing events and steps as well as further information, depending on the messages logged. This could be an EscalationEvent for example, which will trigger an alert showing up in the message processing log.

The log header shows the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartTime</td>
<td>Start of message processing.</td>
</tr>
<tr>
<td>StopTime</td>
<td>End of message processing.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OverallStatus</td>
<td>Specifies the end-to-end status of message processing and corresponds to the Status attribute in the Message Monitoring editor. For more information on statuses see Message Status in the Related Information section.</td>
</tr>
<tr>
<td>ChildCount</td>
<td>Indicates the serial number of the current processing step.</td>
</tr>
<tr>
<td>ChildrenCounter</td>
<td>Specifies the total number of message processing steps.</td>
</tr>
<tr>
<td>ContextName</td>
<td>Integration flow name.</td>
</tr>
<tr>
<td>CorrelationId</td>
<td>ID that identifies correlated messages. Messages can be correlated, for example, when different integration flows on the same tenant communicate with each other. A correlation ID is a base64-encoded ID that is generated in this case by the first integration flow and stored in the message header. As part of the message header, the CorrelationId is then propagated across all related integration flows (that are in charge of processing the correlated messages).</td>
</tr>
<tr>
<td>CustomHeaderProperties</td>
<td>Can only be added by defining a script in the script API and the properties are displayed in the message processing log header. For more information see Define Scripts in the Related Information section.</td>
</tr>
<tr>
<td>Id</td>
<td>Is displayed in the MPL header if the ID has been previously defined as header property. For more information see Headers and Exchange Properties in the Related Information section.</td>
</tr>
<tr>
<td>IntermediateError</td>
<td>True if an error occurred (even temporarily) during message processing, or message processing took more than 1 minute. The header then contains an additional property: LastError showing the error type.</td>
</tr>
<tr>
<td>MessageGuid</td>
<td>Key that identifies the message uniquely in the database.</td>
</tr>
<tr>
<td>MessageType</td>
<td>Is displayed in the MPL header if the message type has been previously defined as header property. For more information see Headers and Exchange Properties in the Related Information section.</td>
</tr>
<tr>
<td>Node</td>
<td>Host name of the node that processed the message.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ReceiverId</td>
<td>Is displayed in the MPL header if the receiver ID has been previously defined as header property. For more information see Headers and Exchange Properties in the Related Information section.</td>
</tr>
<tr>
<td>SenderId</td>
<td>Is displayed in the MPL header if the sender ID has been previously defined as header property. For more information see Headers and Exchange Properties in the Related Information section.</td>
</tr>
<tr>
<td>LocalSuccessor Id</td>
<td>Is set during aggregation and incoming message processing logs get the aggregated MPL as a successor.</td>
</tr>
<tr>
<td>LocalPredecessor Id</td>
<td>Is set during creation of a new MPL, if the message header SAP_MessageProcessingLogID is present or the message header SapPredecessorMessageProcessingLogId is set.</td>
</tr>
</tbody>
</table>
The log steps show the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch</td>
<td>Indicates that the following steps have been processed in the same branch. If a Split or Multicast Step is used, the steps belonging to one sub-message are grouped together within one Branch.</td>
</tr>
<tr>
<td>ModelStepId</td>
<td>This ID is used to specify relation between modeled step (in integration flow) and MPL entry. The integration flow model steps are fragmented in the Camel runtime environment into several processing steps. For more information see Error Classification in the Related Information section.</td>
</tr>
</tbody>
</table>
### Property Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StepId</td>
<td>ID of the related integration flow step. It is assigned by the Camel framework.</td>
</tr>
</tbody>
</table>

**i Note**

- The content of the message processing log varies according to the log level you have set.
- The step properties displayed may vary from step to step, according to the message processing.
- The payload is displayed with header and content. You can also download it by clicking on the Download button on top right corner. This provides a zip file containing one header and one payload file.

**→ Remember**

- All string values are limited to 1000 characters. Any string value longer than 1000 characters is cut off after 1000 characters.
- The MPL property names are limited to 100 characters. Any property name longer than 100 characters is cut off after 100 characters.

You can create attachments to the message processing log by using the Script API.

**i Note**

The amount of MPL attachments which can be written is limited to 1 GB per 24 hours. If the limit is reached, MPL attachments will no longer be stored until the amount of MPL attachments written in the last 24 hours is again below 1 GB.

### Related Information

- Setting Log Levels [page 37]
- Monitoring Message Processing [page 25]
- Message Status [page 29]

### 4.1.4 Setting Log Levels

The log level for the message processing log specifies the granularity of information collected by the message processing log.

**→ Remember**

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.
You can set the log level for an integration flow scenario in the Manage Integration Content page of the Web-based Monitor.

Select your integration flow in the Manage Integration Content and set the log level for this integration flow in the Log Configuration section.

You can choose the following log levels:

- **None**: No Data is recorded during message processing and no data is shown in data monitoring.
- **Info**: Basic information is recorded during message processing. The header is always displayed and in case of failed messages, additional detailed information about the last 50 steps is retained and displayed in the message processing log.
- **Debug**: Detailed information is recorded for all steps during message processing. The header and additional information about the last 100 steps are displayed in the message processing log.
- **Trace**: Detailed information is recorded for all steps and in addition, the message content is tracked. The trace function expires after a certain time (default value: 10 minutes). After expiry the log level switches back to the log level set before. The recorded message content is also retained for a certain time (default value: 1 hour).

**Note**

- As the log level Trace is a high resource consuming feature, SAP can disable this function for a specific tenant, or adjust expiry and retention time.
- If the downloaded trace data for adapter and integration flow steps is more than 25MB then it will be truncated.
- If the downloaded trace data, header value size and exchange properties are more than 10000 characters then it will be truncated.
- If the downloaded trace data header key size and exchange properties are more than 1000 characters then it will be truncated.
- If the downloaded trace data headers and exchange properties are more than 300 then it will be truncated.

**Note**

If you update an integration flow, that is, deploy an integration flow without deleting the predecessor version, the log level configuration is kept stable. If you undeploy an integration flow and then deploy a newer version, it is treated as a completely new integration flow which is logged with log level Info. Changes to the log level do not affect messages already in process and only apply to newly created messages after the time of modification.

**Remember**

We recommend to set log level **Debug** and **Trace** only in a development/test environment!

**Related Information**

_message Processing Log - Text View [page 33]_
Monitoring Message Processing [page 25]
4.2 Managing Integration Content

The Manage Integration Content section provides an overview of integration content artifacts, such as integration flows, that have been deployed on the tenant.

→ Remember

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.

In the Manage Integration Content area, a set of tiles is displayed that show the number of deployed integration content artifacts for a particular type (for example, Integration Flow) and with a particular status (for example, Started). You can configure each tile to filter by a different artifact type or status, or both. To do this, position the cursor on a tile and choose Edit from the context menu.

To open the integration content monitor for a particular set of artifacts (as defined by the tile filter), click a tile.

When you click a tile, a list of integration artifacts is displayed (for the filter settings as defined for the tile). On the left side of the screen, the list of artifacts is shown in a table with the following attributes for each artifact:

- Name
  - Below the name, the artifact type (for example, Value Mapping) is shown.
- Status

To customize the list of displayed artifacts, you can either search for specific artifacts by artifact name or ID or you can filter the table content by attributes such as Status or Type.

To sort and filter the content of the table, choose Table Settings. On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

On the right side of the screen, more details about the artifact selected on the left side are shown. Depending on the artifact type, other functions can be accessed as well.

Integration Content Details

The header area provides the following information about the selected artifact:

- **Deployed on**
  - Time when the artifact was deployed.
- **Deployed by**
  - User who deployed the artifact.
- **Version**
  - Version of the artifact, for example, the integration flow version.
- **ID**
• **Package**
  Name of the package the artifact was deployed from.

• **Mode** (if applicable)
  Indicates whether it is configure-only content.

The header area also provides certain functions (the available functions depend on the type of the selected artifact):

• **Restart**
  Allows you to restart an artifact (for example, an integration flow that has been stopped on the node).

• **Undeploy**

• **Download**
  Allows you to download the artifact to your computer.
  You **cannot** configure-only.

Below the header, the following sections contain detailed information about the selected artifact:

• **Endpoints**
  Shows the URLs of the services exposed by the artifact for a sending application. You can copy the URL by pressing the Copy button.

  For example, if the artifact is an integration flow with a SOAP adapter configured as the sender channel, the endpoint URL specifies how the service exposed by the integration flow can be reached by a SOAP client.

  **Note**
  If the artifact is an OData service or an integration flow with an OData adapter configured as the sender channel, the endpoint URL is currently not available. You can construct the endpoint URL in the following format:

  `<IFLMAP URL>/gw/odata/<OData Service Namespace>/<OData Service Name>;v=1`

  The following table lists the main segments of the endpoint URL.

<table>
<thead>
<tr>
<th>Endpoint URL Segment</th>
<th>Description</th>
</tr>
</thead>
</table>
| IFLMAP URL           | Use the following procedure to select the IFLMAP URL:  
  1. If you have set up Integration Operations tooling in Eclipse, start Eclipse and proceed to step 3.  
  2. Install and configure Eclipse. For more information, see .  
  3. In the Node Explorer view, choose Show Properties from the context menu of the IFLMAP node of your tenant. The Properties view appears with the Node tab selected by default.  
  4. In the General area, copy the value in the URL field.  
  This is the IFLMAP URL. |
### Endpoint URL Segment

<table>
<thead>
<tr>
<th>Endpoint URL Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OData Service Namespace</td>
<td>The namespace represents the logical grouping of the OData service. It is available in the metadata information of the OData service artifact. For more information, see.</td>
</tr>
<tr>
<td>OData Service Name</td>
<td>Ensure that the name of the OData service is capitalized in the endpoint URL. You can get the name of the OData service artifact from its metadata information. For more information, see.</td>
</tr>
</tbody>
</table>

### Status Details

Shows the status of the artifact with regard to its usage at runtime (for example, if it is starting). Also shows the Polling Information for all message polling adapters (such as SFTP or mail adapter) in use. The view provides the respective information on:

- the adapter URI,
- the precise date and time when the latest poll was executed,
- whether further polls are scheduled or not (No further polls scheduled/Further polls scheduled),
- the status of the poll (Successful/Failed),
- (if available) a detailed error message.

If an error occurs during the lifecycle of an artifact, detailed information about the error is displayed. You can copy this information to the ticket you open to get the error solved.

### Note

Errors can occur in various steps during the lifecycle of an artifact. If you think of the artifact as an integration flow, steps are executed on the tenant management node (which the user is connected to when deployment of the integration flow is triggered through the Web UI) as well as on the runtime node where the integration flow is processed at runtime.

Let’s look at an example of an error that is related to the runtime node: If deployment of an integration flow is triggered, which requires a runtime node with a specific node profile, but no runtime node with this profile has been started in the cluster, the artifact cannot be distributed from the tenant management node to the runtime node. In this case, an error is displayed together with the information that no suitable runtime node has been started for this use case.

### Artifact Details

The type of the selected artifact determines which kind of information is displayed.

- Artifact Details for integration flows:
  - The link [Monitor Message Processing](#) allows you to navigate to the Monitor Message Processing application.
  - The link [View Integration Flow](#) allows you to navigate to the integration flow model.
- Artifact Details for value mappings
  - The link [View Value Mapping](#) allows you to navigate to the value mapping design screen.
- Artifact Details for OData services:
  - Name, Namespace, and Version of the OData service are displayed.
- Artifact detail for Integration Adapter:
  - Vendor
4.2.1 Artifact Statuses

An integration content artifact can have different statuses for each node.

The following statuses are possible.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STARTED</td>
<td>The artifact is ready to be used on this node. An artifact with this status is in a final state.</td>
</tr>
<tr>
<td>ERROR</td>
<td>The artifact requires attention by the user (administrator). An artifact with this status is in a final state.</td>
</tr>
<tr>
<td>STARTING</td>
<td>The artifact is in the process of being deployed on the node(s) and, if deployed on the node(s), is being started. An artifact with this status is in an intermediate state.</td>
</tr>
<tr>
<td>STOPPING</td>
<td>The artifact is in the process of being stopped on the node(s) and, if stopped, is being undeployed. An artifact with this status is in an intermediate state.</td>
</tr>
</tbody>
</table>

The status is determined for each node (for example, the status is STARTED on two runtime nodes and ERROR on one runtime node).

Status information is displayed in the format: \texttt{status(number of nodes)}. Different statuses for the nodes are separated with a space.

4.2.2 Endpoint Visualization

You can visualize the adapter endpoints in the \texttt{Manage Integration Content} section.

With the new release of integration service, the endpoint definitions of some adapters are extracted.

Go to \texttt{Manage Integration Content} and select an integration flow from the list and go to \texttt{Endpoints} in the detailed view.

You can visualize the endpoint URL and the definition as specified in the table below, copy the URL to the clipboard or download the definition.
For the SOAP sender adapter, you have 2 options to download the WSDL. You can download the WSDL with or without policies. The WSDL with policies includes WS-policy assertions defining special requirements the sender system must adhere to. If the sender system does not understand WSDLs with policies, the WSDL without policies can be used instead.

Example:

<table>
<thead>
<tr>
<th>Adapter</th>
<th>Show Endpoint (access) URL</th>
<th>Show Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IDOC</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>HTTP</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>AS2</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XI Adapter</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

4.3 Managing Security

The Manage Security section allows you to manage various kinds of security material (for example, user credentials), keystore entries, certificate-to-user mappings and to perform outbound connectivity tests.

Related Information

- Managing Security Material [page 44]
- Managing Keystore Entries [page 56]
- Managing Certificate-to-User Mappings [page 83]
- Performing Connectivity Tests [page 97]
4.3.1 Managing Security Material

The Manage Security Material area provides an overview of security-related artifacts.

You open the Manage Security Material area with the following actions:

Click the tile Security Material.

Security artifacts with the corresponding status (of the tile) are displayed.

Security Material Overview

A list of security material is displayed in a table. For each artifact, the following attributes are displayed:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Display name of the artifact</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td>• User Credentials</td>
</tr>
<tr>
<td></td>
<td>For User Credentials artifacts (a tooltip indicates the kind of the artifact).</td>
</tr>
<tr>
<td></td>
<td>• OAuth2 Credentials</td>
</tr>
<tr>
<td></td>
<td>• OAuth2 Authorization Code</td>
</tr>
<tr>
<td></td>
<td>• Secure Parameter</td>
</tr>
<tr>
<td></td>
<td>For Secure Parameter artifacts (a tooltip indicates the kind of the artifact).</td>
</tr>
<tr>
<td></td>
<td>• Known Hosts (SSH)</td>
</tr>
<tr>
<td></td>
<td>• PGP Public Keyring</td>
</tr>
<tr>
<td></td>
<td>• PGP Secret Keyring</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Displays the state with regard to the artifact deployment.</td>
</tr>
<tr>
<td></td>
<td>This status indicates whether an artifact has been deployed successfully on a</td>
</tr>
<tr>
<td></td>
<td>tenant or whether it still needs to be authorized.</td>
</tr>
<tr>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td>• Stored</td>
</tr>
<tr>
<td></td>
<td><strong>iNote</strong> Refresh the list to see the current status.</td>
</tr>
<tr>
<td></td>
<td>• Deployed</td>
</tr>
<tr>
<td></td>
<td>• Error</td>
</tr>
<tr>
<td></td>
<td>• Unauthorized (for OAuth2 Authorization Code)</td>
</tr>
</tbody>
</table>
### Attribute | Description
--- | ---
Deployed By | User who deployed the artifact
Deployed On | Time when the artifact was deployed

To sort and filter the content of the table, choose **Table Settings**. On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

### Actions

To create/deploy a new artifact, choose **Create** and select the artifact type (possible for the following artifact types: User Credentials, OAuth2 Client Credentials, OAuth2 SAML Bearer Assertion, OAuth2 Authorization Code and Secure Parameter).

To upload a new artifact, choose **Upload** and select the artifact type (possible for the following artifact types: Known Hosts (SSH), PGP Public Keyring, PGP Secret Keyring).

To authorize an existing artifact, select the artifact in the table and choose **Authorize** (only supported for OAuth2 Authorization Code).

To edit and redeploy an existing artifact, select the artifact in the table and choose **Edit** (only supported for Credentials).

To download an artifact, select the artifact in the table and choose **Download** (not available for Credentials).

You can also delete an artifact (supported for all artifact types except Keystore).

### Related Information

- Deploying or Editing a User Credentials Artifact [page 46]
- Deploying an SSH Known Hosts Artifact [page 54]
- Deploying a Secure Parameter Artifact [page 53]
- Deploying an OAuth2 Client Credentials Artifact [page 47]
- Deploying a PGP Secret Keyring [page 55]
- Deploying a PGP Public Keyring [page 54]
4.3.1.1 Deploying or Editing a User Credentials Artifact

To set up a connection using basic authentication or username token authentication, you have to specify the required attributes (for example, user name and password).

Context

User Credentials (security artifact) on the corresponding tenant. You can specify basic authentication either for a connected SuccessFactors system (through the SuccessFactors adapter) or for general SOAP connectivity or an OpenConnectors service (through the OpenConnector adapter). To enable basic authentication for a runtime node, you specify the user credentials and deploy the attributes.

**Note**

You can also edit and redeploy an existing artifact. To do that, select the artifact in the table under Manage Security Material and choose Edit.

Procedure

1. Click the tile in the Manage Security Material section.
2. Click Create.
3. Specify the following properties:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A name for the artifact</td>
</tr>
<tr>
<td>Description</td>
<td>A description of the artifact (optional)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>This allows you to configure a specific system to enable a connection with your integration flow artifact.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If you select:</td>
</tr>
<tr>
<td></td>
<td>○ SuccessFactors: Specify the Company ID (which indicates the client instance used to connect to the</td>
</tr>
<tr>
<td></td>
<td>SuccessFactors system).</td>
</tr>
<tr>
<td></td>
<td>○ OpenConnectors: Specify the Organization and Element (value obtained from the authorization</td>
</tr>
<tr>
<td></td>
<td>header that helps to authenticate with the specific instance).</td>
</tr>
<tr>
<td></td>
<td>See:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User</th>
<th>The user that calls the receiver system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>Password against which the user has to be authenticated</td>
</tr>
</tbody>
</table>

Results

When you refresh the Manage Security Material page, the new artifact is displayed (with Type Credentials) in the artifact table.

Next Steps

**i Note**

When you have a User Credentials artifact, you also have to re-deploy and restart all integration flows that use this artifact.

Related Information

Managing Security Material [page 44]

**4.3.1.2 Deploying an OAuth2 Client Credentials Artifact**

Many web servers can use OAuth 2.0 for authorization purposes. If you want to connect to a system that uses OAuth 2.0 authentication, you need to deploy an OAuth2 Credentials artifact using the following procedure.

Context

You can edit and deploy an OAuth2 Client Credentials artifact.

**i Note**

Note that every time you edit an OAuth2 Credentials artifact, have to re-enter the Client Secret.


Procedure

1. Click the tile in the Manage Security Material section.
2. Choose Create OAuth2 Client Credentials.
3. Specify the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the artifact that you want to deploy on the tenant.</td>
</tr>
</tbody>
</table>
| Grant Type          | The relevant type of grant for authorizing the client to interact with the server. The available grants are:  
|                     | ○ Client Credentials  
|                     | ○ OAuth2SAMLBearerAssertion                                                  |
| i Note              | Read the blog, to understand how to design and deploy an integration flow that talks to SuccessFactors OData V2 endpoint with OAuth2 authentication. |
| Description         | Description of the artifact name you are deploying on the tenant.           |
| Authentication URL  | URL of the OAuth2 authorization server that issues the access token.         |
| Client ID           | ID of the client that you are connecting to.                                |
| Client Secret       | Secret key of the client that you are connecting to.                        |
| Client Authentication| OAuth2 uses a multiple step authentication pattern: Client credentials (Client ID and Client Secret, as specified in the artifact) are used by the client application to initially request an access token. The access token is then used to authorize the client (for as long as the token is valid) to access the server’s resources (for example, the resources that are used in the associated integration flow). In many OAuth2 scenarios, the access token is issued (or generated) by an authorization server. |
| Include Scope       | Allows you to access an application using Client ID and Client Secret.     |
|                     | By default the Send as Body Parameter is selected, this option sends the Client ID and Client Secret as a json content to the authentication server in the request body. If you select the Send as Request Header option, then the Client ID and Client Secret are encoded, and send to the server as an Authorization header. |
|                     | Provide the OAuth2 scope information to be included in the request body.   |
## Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Type</td>
<td>Request content type to indicate the media type.</td>
</tr>
</tbody>
</table>

4. Choose **Deploy**.

### Related Information

- Managing Security Material [page 44]

### 4.3.1.3 Deploying an OAuth2 SAML Bearer Assertion

Many web servers can use OAuth 2.0 for authorization purposes. If you want to connect to a system that uses OAuth 2.0 authentication, you need to deploy an OAuth2 Credentials artifact using the following procedure.

### Context

You can edit and deploy an OAuth2 Credentials artifact.

**Note**

Note that every time you edit an OAuth2 Credentials artifact, you have to re-enter the Client Secret.


### Procedure

1. Click the tile in the Manage Security Material section.
2. Choose **Create** ➔ **OAuth2 SAML Bearer Assertion**.
3. Specify the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the artifact that you want to deploy on the tenant</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Grant Type</td>
<td>The relevant type of grant for authorizing the client to interact with the server. The available grants are:  ○ <em>Client Credentials</em>  ○ <em>OAuth2SAMLBearerAssertion</em></td>
</tr>
<tr>
<td>i Note</td>
<td>Read the blog, to understand how to design and deploy an integration flow that talks to SuccessFactors OData V2 endpoint with OAuth2 authentication.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the artifact name you are deploying on the tenant</td>
</tr>
<tr>
<td>Audience</td>
<td>Provide the host name of the target system, to which you want to establish the connection.</td>
</tr>
<tr>
<td>Client Key</td>
<td>A unique identifier created by the target system to identify the client.</td>
</tr>
<tr>
<td>Token Service URL</td>
<td>Provide the URL that generates OAuth2 token for the registered OAuth2 client.</td>
</tr>
<tr>
<td>Target System Type</td>
<td>Specify the relevant host system for authenticating the user against the system. Select one of the following system:  ○ SuccessFactors  ○ SAP Cloud Platform (Neo)  ○ SAP Cloud Platform (CF)</td>
</tr>
<tr>
<td>Company ID</td>
<td>Specify the company ID of your SuccessFactors instance. This parameter is available only if you select target system type as SuccessFactors.</td>
</tr>
<tr>
<td>Token Service User</td>
<td>Username required to access the <em>Token Service URL</em>. (only if you select target system type as SAP Cloud Platform)</td>
</tr>
<tr>
<td>Token Service Password</td>
<td>Password required to access the <em>Token Service URL</em>. (only if you select target system type as SAP Cloud Platform)</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Additional Properties</td>
<td>Provide the Key and Value parameters of to fetch the access token.</td>
</tr>
</tbody>
</table>

**i Note**

Maintain the Value of the key field `userIdSource` for each environment as follows:
- If your tenant is on Neo environment, set the value as `mail`
- If your tenant is on Cloud Foundry environment, set the value as `email`

4. Choose **Deploy**.

**Related Information**

*Managing Security Material [page 44]*

**4.3.1.4 Deploying an OAuth2 Authorization Code**

Many web servers can use OAuth 2.0 for authorization purposes. If you use an OAuth2 Authorization Code, the authentication is done via an authorization server as an intermediary step. The client can exchange the OAuth2 Authorization Code for an access token. Your user credentials are never shared with the client.

**Context**


**i Note**

Currently, only the Provider „Microsoft 365“ is supported.

**Procedure**

1. Click the tile in the *Manage Security Material* section.
2. Choose **Create** > **OAuth2 Authorization Code**.
3. Specify the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the artifact that you want to deploy on the tenant.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the artifact you are deploying on the tenant.</td>
</tr>
<tr>
<td>Provider</td>
<td>Enter the name of the provider of the platform on which you created the OAuth2 client.</td>
</tr>
<tr>
<td>Authorization URL</td>
<td>Provide the Authorization URL for authorizing the OAuth client to access resources of a user.</td>
</tr>
<tr>
<td>Token Service URL</td>
<td>Provide the URL that generates the OAuth2 access and refresh token for the registered OAuth2 client and the provided user.</td>
</tr>
<tr>
<td>Refresh Token Expiry</td>
<td>Provide an amount of time after which the refresh token expires.</td>
</tr>
<tr>
<td></td>
<td>The default is set to 90 days but should be min. 3 days. The SAP Cloud Integration system will update the refresh token before the expiration.</td>
</tr>
<tr>
<td>Client ID</td>
<td>ID of the client you want to connect to.</td>
</tr>
<tr>
<td>Client Secret</td>
<td>Secret key of the client that you are connecting to.</td>
</tr>
<tr>
<td>Send As</td>
<td>○ Body Parameter: Send the Client ID and Client Secret in the request body when calling the Authorization URL or Token Service URL.</td>
</tr>
<tr>
<td></td>
<td>○ Basic Auth Header: Send the Client ID and Client Secret via Basic Authorization when calling the Authorization URL or Token Service URL.</td>
</tr>
<tr>
<td>User Name</td>
<td>Name of the user whose resources the OAuth2 client gets access to.</td>
</tr>
<tr>
<td>Scope</td>
<td>OAuth2 scopes protecting the access to the resources.</td>
</tr>
</tbody>
</table>

**iNote**

*Configure Mail Sender and Receiver Adapter:*

○ The scope https://outlook.office.com/SMTP.Send is necessary for sending e-mails from Outlook 365.

○ The scope https://outlook.office.com/IMAP.AccessAsUser.All is necessary for reading e-mails from Outlook 365.

○ You can also request an offline_access scope. When a user approves the offline_access scope, your app can receive refresh tokens from the Microsoft identity platform token endpoint. If the offline_access scope is not provided, Cloud Integration is adding this scope automatically to the credential since Cloud Integration needs the refresh token to create access tokens.

○ If you add more than one scope, you need to separate your scopes by a blank space.

4. Choose *Deploy*. 

---

52  Operations Guide  Monitoring
4.3.1.5 Deploying a Secure Parameter Artifact

Use the secure parameter artifact to deploy confidential data, for example, for custom adapters.

Context

You can store secure data like passwords in a secure store and use an alias name to access this data in an integration flow. The secure parameter artifact contains this alias name and confidential data.

Procedure

1. Click the tile in the Manage Security Material section.
2. Choose Create Secure Parameter.
3. Specify the following attributes.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Provide a name for the artifact. The artifact name is used as an alias for the confidential data assigned by this parameter.</td>
</tr>
<tr>
<td>Description</td>
<td>Provide a more detailed description of the artifact.</td>
</tr>
<tr>
<td>Secure Parameter</td>
<td>Enter the confidential value of the attribute.</td>
</tr>
<tr>
<td>Repeat Secure Parameter</td>
<td>Repeat the confidential value of the attribute.</td>
</tr>
</tbody>
</table>


Results

When you refresh the Manage Security Material page, the new artifact is displayed in the table.

Related Information

Managing Security Material [page 44]
4.3.1.6 Deploying an SSH Known Hosts Artifact

This artifact type specifies the known hosts file used when configuring secure connectivity based on SSH File Transfer Protocol (SFTP).

Context

The known hosts file contains the public keys and addresses of the trusted SFTP servers.

Procedure

1. Click the tile in the Manage Security Material section.
2. Choose Upload Known Hosts (SSH).
3. Browse to the known hosts file on your computer.

Results

When you refresh the Manage Security Material page, the new artifact is displayed in the table.

Related Information

Managing Security Material [page 44]

4.3.1.7 Deploying a PGP Public Keyring

This artifact contains the public key that enables the tenant to encrypt or verify messages using the Pretty Good Privacy (PGP) standard.

Prerequisites

You have created a PGP keyring file and stored it on your computer.
**Procedure**

1. Select the tile *Security Material*.
2. Go to *Upload* and select the option *PGP Public Keyring*.
3. Browse for the public keyring file on your computer.
4. Click on *Deploy* button to deploy your artifact.

**i Note**

To update the list, click the refresh button.

You can also select an artifact from the list and click *Download*.

To undeploy an artifact select it from the list and click *Undeploy*. The system displays a warning and requires you to confirm the task. The undeployed artifact is removed from the list.

**Related Information**

Managing Security Material [page 44]

### 4.3.1.8 Deploying a PGP Secret Keyring

This artifact contains the public and private key pair for the usage of Open Pretty Good Privacy (PGP). The private key enables the tenant to decrypt or sign messages.

**Prerequisites**

You have created a PGP secret keyring and stored it on your computer.

**Procedure**

1. Select the tile *Security Material*.
2. Go to *Upload* and select the option *PGP Secret Keyring*.
3. Browse for the secret keyring file on your computer and enter the passphrase of the PGP secret keyring.

**i Note**

The password was defined when the PGP secret keyring was created. If the entered password does not match the original one, you cannot deploy the artifact and get an error message.
4. Click the **Deploy** button to deploy your artifact.

<table>
<thead>
<tr>
<th><img src="image" alt="i Note" /></th>
</tr>
</thead>
<tbody>
<tr>
<td>To update the list click the refresh button.</td>
</tr>
</tbody>
</table>

You can also select an artifact from the list and click **Download**.

To undeploy an artifact select it from the list and click **Undeploy**. The system displays a warning and requires you to confirm the task. The undeployed artifact is removed from the list.

**Related Information**

Managing Security Material [page 44]

### 4.3.2 Managing Keystore Entries

The Keystore Monitor allows a tenant administrator to manage the tenant keystore and its entries (X.509 certificates and key pairs).

**Context**

Connections between a Cloud Integration tenant and a remote system can be secured using client certificate authentication (with X.509 certificates). The key pairs and certificates required to implement this authentication option are stored in a keystore.

A keystore typically contains entries that belong to the tenant administrator (customer) and entries that are owned by SAP. Each entry is uniquely identified by an alias.

The operations you can perform on a keystore entry depend on whether it is owned by SAP or the tenant administrator. You (as tenant administrator) can, for example, only delete or back up entries that are owned by the tenant administrator. Vice versa, SAP cannot access or download any keystore entries owned by the tenant administrator.
The GIF shown below gives you a brief overview of the Keystore tile and its functions:

Keystore entries are identified by an alias.

There are the following entry types:

- **Key Pair** entry
  Consists of a private key and its X.509 certificate chain.
  All private keys of a keystore are encrypted with the same password. This password is also used as the keystore password (for checking the integrity of the keystore). The keystore is never stored in the same database as the encrypted/signed application data. The password is stored in a separate database. The certificate chain typically consists of the public key certificate and the intermediate certification authority (CA) certificate with which the signature of the public key certificate can be verified.

- **Certificate** entry
  In many cases this is an X.509 root certificate.

More information: Tenant Keystore [page 233]

⚠️ **Caution**

Note the following restrictions when managing keystore entries:

The maximum size of a keystore is 2 MB (when using the Neo environment).

- The 2 MB limit corresponds to around 2000 X.509 certificates.
- A key pair with a chain of three X.509 certificates consumes about 3 KB, so if the keystore only contains key pairs of this type, then you can store around 600 key pairs in the keystore.

The maximum size of a keystore is 6 MB (when using the Cloud Foundry environment).

- The 6 MB limit corresponds to around 6000 X.509 certificates.
- A key pair with a chain of three X.509 certificates consumes about 3 KB, so if the keystore only contains key pairs of this type, then you can store around 1800 key pairs in the keystore.
If you upload a whole keystore (.jks file) to the tenant, the maximum keystore size is limited to 2 MB. A locking mechanism prevents different users from changing or deleting entries of the keystore at the same time. If a user is changing the content of a keystore, the keystore is locked for other users during that time. Private keys cannot be downloaded from a keystore.

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. The Current tab shows the keystore entries available for the tenant.

For each keystore, the single entries are displayed (by alias).

The Keystore Monitor shows all keystore entries in a table. For each entry, the following attributes are displayed:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alias</strong></td>
<td>Uniquely identifies the keystore entry.</td>
</tr>
<tr>
<td></td>
<td>A keystore entry can be either be owned by the tenant administrator or by SAP.</td>
</tr>
<tr>
<td></td>
<td>In case a keystore entry it is owned by SAP, the lock icon (Maintained by SAP) is shown next to the alias. SAP-owned keystore entries provide a reduced set of functions. For example, you cannot update or delete an SAP-owned keystore entry.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Indicates whether the entry is a Certificate (X.509 certificate) or a Key Pair (with public and private key and an X.509 certificate chain).</td>
</tr>
<tr>
<td></td>
<td>○ Key Pair</td>
</tr>
<tr>
<td></td>
<td>Consists of a private key and its X.509 certificate chain.</td>
</tr>
<tr>
<td></td>
<td>○ Certificate</td>
</tr>
<tr>
<td></td>
<td>In many cases this is an X.509 root certificate.</td>
</tr>
<tr>
<td><strong>Valid Until</strong></td>
<td>Indicates the expiration date.</td>
</tr>
<tr>
<td></td>
<td>If keys and certificates have expired, the date is highlighted in red.</td>
</tr>
<tr>
<td><strong>Last Modified At</strong></td>
<td>Indicates the date and time the entry was last modified.</td>
</tr>
<tr>
<td><strong>Subject DN</strong></td>
<td>Indicates the Subject distinguished name (DN) which entails information about the owner of the certificate such as the common name of the certificate owner (CN), the organization (O), or the country or region (C).</td>
</tr>
<tr>
<td><strong>Issuer DN</strong></td>
<td>Indicates the Issuer distinguished name (DN) which entails information about the issuer of the certificate such as common name of the certificate issuer (CN), the organization (O), or the country or region (C).</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Actions</td>
<td>Choose the  (Actions) button for a key pair or certificate to perform an action. The available actions depend on the type of the keystore entry and whether it is owned by SAP.</td>
</tr>
<tr>
<td></td>
<td>○ Rename</td>
</tr>
<tr>
<td></td>
<td>○ Update</td>
</tr>
<tr>
<td></td>
<td>○ Download</td>
</tr>
<tr>
<td></td>
<td>○ Delete</td>
</tr>
<tr>
<td></td>
<td>○ Download Certificate</td>
</tr>
<tr>
<td></td>
<td>○ Download Public OpenSSH Key</td>
</tr>
<tr>
<td></td>
<td>○ Update Signing Response</td>
</tr>
<tr>
<td></td>
<td>○ Download Certificate Chain</td>
</tr>
<tr>
<td></td>
<td>○ Download Root Certificate</td>
</tr>
<tr>
<td></td>
<td>○ Download Signing Request</td>
</tr>
</tbody>
</table>

**Note**

SAP-owned entries are indicated by a lock icon. You cannot change, delete, or back them up.

**Next Steps**

The Keystore Monitor also gives you the following options:

- Showing details of a keystore entry
- Creating a Key Pair or a SSH Key Pair
- Uploading a keystore or SSH key
  You can upload or add individual entries to an existing keystore. In the latter case, you can overwrite existing entries or keep them.
  SAP-owned keystore entries are indicated by a lock icon. You cannot change or delete them.
- Downloading the public content of a keystore or a single keystore entry
- Deleting a keystore entry
- Backing up keystore entries owned by the tenant administrator
- Restoring backed-up keystore entries
- Downloading backed-up keystore entries

**Related Information**

- Displaying Properties of a Keystore Entry [page 60]
- Importing a Certificate [page 63]
- Importing a Key Pair [page 64]
- Importing a Keystore [page 66]
- Changing the Alias of a Keystore Entry [page 68]
- Updating a Key Pair [page 68]
### 4.3.2.1 Displaying Properties of a Keystore Entry

Display properties of a selected keystore entry.

**Context**

There are the following entry types:

- **Key Pair** entry
  Consists of a private key and its X.509 certificate chain.
  All private keys of a keystore are encrypted with the same password. This password is also used as the keystore password (for checking the integrity of the keystore). The keystore is never stored in the same database as the encrypted/signed application data. The password is stored in a separate database.
  The certificate chain typically consists of the public key certificate and the intermediate certification authority (CA) certificate with which the signature of the public key certificate can be verified.

- **Certificate** entry
  In many cases this is an X.509 root certificate.

**Procedure**

1. Choose the **Keystore** tile in the **Manage Security** section.
2. Click the alias of a keystore entry to show the details.
3. The following attributes are shown for the selected keystore entry.
In case, the keystore entry is part of a certificate chain, the certificate chain is displayed as a hierarchy tree at left.

The following details are displayed for the selected keystore entry. When you click a certain element in the certificate chain, the details are adapted to show the details of this element.

- The Keystore attribute indicates if the displayed entry is part of the currently active keystore (Current) or of the backed-up keystore (Backup).
- The Alias is used to refer to a specific public key from a keystore.
- The Owner can either by SAP or Tenant Administrator.

### Administration Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created By</td>
<td>User who created the certificate</td>
</tr>
<tr>
<td>Created On</td>
<td>Time when certificate has been created</td>
</tr>
<tr>
<td>Last Modified By</td>
<td>User that modified the keystore entry at the latest</td>
</tr>
<tr>
<td>Last Modified On</td>
<td>Time when the keystore entry has been modified at the latest</td>
</tr>
</tbody>
</table>

**Note**

The user shown in Last Modified By is anonymized (the string SAP is shown) in case the keystore is owned by SAP.

### General Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>○ <strong>Key Pair</strong>&lt;br&gt;Consists of a private key and its X.509 certificate chain.&lt;br&gt;○ <strong>Certificate</strong>&lt;br&gt;In many cases this is an X.509 root certificate.</td>
</tr>
<tr>
<td>Version</td>
<td>Certificate version</td>
</tr>
<tr>
<td>Serial Number</td>
<td>Serial number of certificate as provided by the issuer</td>
</tr>
<tr>
<td>Subject DN</td>
<td>Distinguished name of the entity associated with the public key of the certificate</td>
</tr>
<tr>
<td>Issuer DN</td>
<td>Distinguished name of the issuer (the certification authority that issued and signed the certificate)</td>
</tr>
<tr>
<td>Key Type</td>
<td>Cryptographic standard the keystore entry is using (for example, RSA)</td>
</tr>
<tr>
<td>Key Size</td>
<td>Number of bits (also referred to as Key Length)</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Signature Algorithm</td>
<td>Algorithm used to create the signature of the key entry (for example, SHA-512)</td>
</tr>
<tr>
<td>Valid From</td>
<td>Start of validity period</td>
</tr>
<tr>
<td>Valid Until</td>
<td>End of validity period</td>
</tr>
</tbody>
</table>

The section *Fingerprints* shows different sequences defined to identify the public key. A fingerprint is generated out of the public key applying a hash function on the public key. For each fingerprint, the hash algorithm is also displayed (for example, SHA-512).

**Next Steps**

The following functions are available:

- You can delete the keystore entry in case it is owned by the Tenant Administrator.
- You can download the keystore entry.
- You can change the alias of the keystore entry in case it is owned by the Tenant Administrator.

**Related Information**

*Changing the Alias of a Keystore Entry* [page 68]

**4.3.2.2 Creating a Key Pair/SSH Key Pair**

Create a key pair or a Secure Shell (SSH) key pair.

**Context**

You create a key pair to use it for SSL, decryption, signature, and client certificate authentication.

You create a SSH key pair to connect to the SFTP server.
Procedure

1. Choose the Keystore tile in the Manage Security section.
2. On the Current tab, choose Create Key Pair (above the table).
   - If you like to create an SSH key (as required, for example, for the communication with SFTP servers), choose Create SSH Key.
3. In the next screen, enter the required information.
   - For the Alias, enter a string that is used to identify the keystore entry. Note that the alias has to be unique per tenant. If you try to create a keystore entry with an alias that already exists on the tenant, an error message is displayed.
   - Select appropriate settings for Key Type, Key Size and Signature Algorithm.
   - When creating an SSH key, for Key Type the options RSA and EC (Eliptic Curve) are supported. When creating other keys, the options RSA, DSA and EC are supported.
   - Enter the required information for the certificate attributes.
     - These attributes represent the coordinates of the organization associated with the certificate (such like Organization (O), Organizational Unit (OU), Country (C), and so forth).
     - Note that for Common Name (CN) you need to specify an address (for example, myhost.com). Which address to specify, depends on the usage of the key pair. For example, if you use the key pair to sign a message, the address of the signing component is to be specified.
   - For more information on certificates, see: X.509 Certificates [page 233]
4. Choose Create.

Related Information

Importing a Key Pair [page 64]
Updating a Key Pair [page 68]

4.3.2.3 Importing a Certificate

Import a single certificate to the tenant keystore.

Context
**Procedure**

1. Choose the **Keystore** tile in the **Manage Security** section.
2. On the **Current** tab, above the table, choose **Add**, then **Certificate**.
3. Enter the alias (**Alias** field).

   **Note**
   
   You cannot import a certificate with an alias starting with `sap_` (as this namespace is reserved for SAP-owned keystore entries).
   
   Make sure you don’t enter an alias that is already used by an existing keystore entry.

4. Choose **Browse** and select the certificate on your local disk.
5. Choose **Add**.

   **Note**
   
   If there is already an entry with this alias in the keystore, an error message is shown.
   
   Currently, the size limit for uploading certificates to the keystore is at 10240 bytes.
   
   If the system cannot verify the trustworthiness of the certificate (for example, it cannot verify its signature), a warning message is displayed, which also contains the fingerprints.
   
   A fingerprint is generated out of the public key applying a hash function to it. You can check the trustworthiness of the public key by comparing the fingerprint with a fingerprint for the public key provided by your communication partner via a trustworthy channel (phone, signed e-mail).

**4.3.2.4 Importing a Key Pair**

Import a private/public key pair to the tenant keystore.

**Context**

**Note**

You cannot import a key pair for a keystore entry with an alias starting with `sap_` or `hcicertificate` (as these entries are owned by SAP).

**Procedure**

1. Choose the **Keystore** tile in the **Manage Security** section.
2. On the **Current** tab, above the table, choose **Add Key Pair**.

3. Enter the alias (**Alias** field).

   **Note**
   
   You cannot import a key pair with an alias starting with (as this namespace is reserved for SAP-owned keystore entries). You cannot import a key pair with an alias starting with `sap`. You cannot import a key pair with an alias starting with (as this namespace is reserved for SAP-owned keystore entries). You cannot import a key pair with an alias starting with
   
   Make sure you don't enter an alias that is already used by an existing keystore entry.

4. Choose **Browse** and select the key pair on your local disk.

   You can also upload files with the extension `.pfx` and `.p12`.

   **Caution**
   
   Note that a `.pfx` or `.p12` file can contain one or more keys. In case the file contains only one key, you can upload it as single key pair with this feature. If the file contains more than one key, it is recommended to upload it as keystore (via **Add Keystore**). The reason is: If you try to upload a `.pfx` or `.p12` file that contains more than one key via **Add Key Pair**, the system will only upload one key and ignore the others.

5. Enter the password associated with the private key.

6. Choose **Add**.

   **Note**
   
   If there is already an entry with this alias in the keystore, an error message is shown.

   Currently, the size limit for uploading key pairs to the keystore is at 30720 bytes.

   If the system cannot verify the trustworthiness of the key pair (for example, if it is self-signed), a warning message is displayed, which also contains the fingerprints.

   A fingerprint is generated out of the public key applying a hash function to it. You can check the trustworthiness of the public key by comparing the fingerprint with a fingerprint for the public key provided by your communication partner via a trustworthy channel (phone, signed e-mail).
4.3.2.5 Importing a Keystore

Upload a new keystore. You can replace an existing keystore completely or update an existing one by adding new entries. In the latter case, you can overwrite existing entries or keep them.

Context

i Note
You cannot upload keystore entries with aliases starting with sap_or hcicertificate. These aliases are reserved for SAP-owned keystore entries.

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. On the Current tab, above the table, choose Add Keystore.
3. Choose Browse and select the keystore on your local disk.

Upload of keystores supports the following formats:

- JKS format (file extension .jks) for Java keystores
- PKCS#12 format (file extension .pfx or .p12)

A .pfx or .p12 file can contain one or more keys. In case the file contains only one key, you can also upload it as single key pair (via Add Key Pair).

After the import of a .pfx or .p12 file, the root certificates of all keys contained in the keystore will be displayed separately in the keystore overview (in Web UI Monitoring under Manage Security in the Keystore tab).

4. Enter the keystore password.
5. In the Action field, select one of the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds the entries from the uploaded keystore so that they are merged with the ones in the existing keystore.</td>
</tr>
<tr>
<td>Replace</td>
<td>Replaces the whole keystore with the uploaded one.</td>
</tr>
</tbody>
</table>

6. If you have selected Add (in the Action field), you can specify whether existing entries are to be overwritten by uploaded entries with the same alias.

i Note
Exception: SAP-owned entries are preserved as they cannot be changed or deleted by the tenant administrator.
7. Choose Deploy.

Next Steps

A dialog is displayed showing a summary of the added, preserved, and overwritten entries. If any entries could not be uploaded, information is provided to explain why (for example, you tried to upload a keystore entry with an alias starting with sap).

4.3.2.6 Importing an SSH Key

Import an SSH key or a putty key to the tenant keystore.

Context

It is possible to add an SSH or putty key to the keystore monitor instead of creating one. If you want to learn more about how to setup connections to the sftp servers, go and visit this blog: Cloud Integration - How to Setup Secure Connection to sftp Server.

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. On the Current tab, above the table, choose Add SSH Key.
3. Enter an alias for the keystore entry (Alias field).
4. Choose Browse and select the SSH key or putty key on your local disk.
5. Specify the password for the key and define key specific values.
6. Enter the required information for the certificate attributes.
   These attributes represent the coordinates of the organization associated with the certificate (such like Organization (O), Organizational Unit (OU), Country (C), and so forth).
   Note that for Common Name (CN) you need to specify an address (for example, myhost.com). Which address to specify, depends on the usage of the key pair. For example, if you use the key pair to sign a message, the address of the signing component is to be specified.
   For more information on certificates, see: X.509 Certificates [page 233].
7. Check the settings for the validity (Valid From and Valid Until fields) and, if desired, change them (by default, the validity is set to 10 years).
8. Choose Add to create the key.
4.3.2.7 Changing the Alias of a Keystore Entry

Context

An alias is a reference to a single keystore entry. You can use an alias to refer to and select a specific public key from a keystore.

Note

- You can only change the alias of a keystore entry owned by the Tenant Administrator.
- The maximal length of a keystore alias is 250 characters.
- You cannot use aliases that are reserved for SAP (for example, starting with `sap`).

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. Click the alias of a keystore entry to show the details.
3. Choose Rename.
4. Enter the new alias and choose Save.

Next Steps

Alternatively, you can change the alias of a keystore entry in the following way: Choose the Keystore tile in the Manage Security section and for a keystore entry click the (Actions) icon and choose Rename.

4.3.2.8 Updating a Key Pair

Update a key pair keeping the alias of the keystore entry unchanged.

Context

This option is not available for keystore entries owned by SAP.
Procedure

1. Choose the Keystore tile in the Manage Security section.
2. On the Current tab, select a key pair, choose the (Actions) icon, then select Update. Alternatively, you can click the key pair alias to open the key pair details, and then choose Update Key Pair.
3. Browse to the key pair file on your computer and enter the password (as the key pair also contains a private key that is protected by a password).

   **i Note**

   You cannot change the alias.

4. Choose Update.

   The key pair is updated in the keystore.

4.3.2.9 Updating a Key Pair with a Signing Response

Upload a signing response from a certification authority and use it to update the key pair in your keystore, keeping the alias of the keystore entry unchanged.

Prerequisites

You have created a certificate signing request (CSR) for the key pair and used this to request a signed certificate from a certification authority (CA). The certification authority has provided a signing response.

Context

This option is not available for keystore entries owned by SAP.

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. On the Current tab, select the key pair that the signing response was created for.
3. Choose the (Actions) icon, then select Update Signing Response. Alternatively, you can click the key pair alias to open the key pair details, and then choose Update Signing Response.
4. Browse to the signing response file that you want to upload.
The following signing response formats are supported:

- PKCS#7 in binary or PEM or base64 textual encoding (*.p7c)
- Chain of DER-PEM encoded X.509 certificates in one file (*.*)
- X.509 PKI path (sequence of certificates with top CA certificate at index 0) (*.pkipath)
- Software Publisher Certificate (SPC) (*.spc)

5. Choose Update.

Current, the upload size limit for signing responses is at 30720 bytes.

Results

The key pair is updated in the keystore. The key pair can now be used to set up secure connections to external receiver back ends.

Related Information


4.3.2.10 Updating a Certificate

Update a certificate keeping the alias of the keystore entry unchanged.

Context

This option is not available for keystore entries owned by SAP.

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. On the Current tab, select the certificate.
3. Choose the (Actions) icon, then select Update. Alternatively, you can click the certificate alias to open the certificate details, and then choose Update.

4. Browse for the certificate file on your computer.
   You cannot change the alias.

5. Choose Update.
   The new certificate is added to the keystore entry.

4.3.2.11 Downloading a Keystore

Download a keystore to your local disk.

Context

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. On the Current tab, choose Download (above the table).

Next Steps

The public part of the keystore is downloaded (in a file with the name PublicContentKeystore.jks).

The file is not password protected. You can open it with a third-party keystore editor (for example, KeyStore Explorer).

For private key entries, the certificate chains are resolved into single entries. The alias of the public key certificate is the same as the alias of the original key-pair entry. The intermediate certificate gets an additional suffix _1 in the alias name. The root certificate gets an additional suffix _2.
4.3.2.12 Downloading Single Keystore Entries

Download a single keystore entry to your local disk.

**Context**

You can download a key, a single certificate, a complete certificate chain, a root certificate, or a signing request (depending on the keystore entry type).

<table>
<thead>
<tr>
<th>i Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can download only public keystore content (no private keys).</td>
</tr>
</tbody>
</table>

**Procedure**

1. Choose the **Keystore** tile in the **Manage Security** section.
2. On the **Current** tab, select a keystore entry.
3. Choose the (Actions) icon, then select one of the options listed in table below. Alternatively, you can click the keystore entry alias to open the details, and then choose the option under **Download**.

The options available depend on the type of keystore entry you have selected.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Pair entry</td>
<td>You can download the public key (X.509 format).</td>
</tr>
<tr>
<td></td>
<td>○ Download Certificate</td>
</tr>
<tr>
<td></td>
<td>A file with the name <code>&lt;alias&gt;.cer</code> is downloaded.</td>
</tr>
<tr>
<td></td>
<td>○ Download Certificate Chain</td>
</tr>
<tr>
<td></td>
<td>A file with the name <code>&lt;alias&gt;.p7b</code> is downloaded.</td>
</tr>
<tr>
<td></td>
<td>The file contains the whole certificate chain.</td>
</tr>
<tr>
<td></td>
<td>○ Download Public OpenSSH Key</td>
</tr>
<tr>
<td></td>
<td>Downloads the public key in OpenSSH format.</td>
</tr>
<tr>
<td></td>
<td>You can store the downloaded <code>&lt;alias&gt;.pub</code> file on the connected SFTP server to enable public key authentication on the server.</td>
</tr>
<tr>
<td>i Note</td>
<td>To configure public key authentication on certain SFTP servers, public keys need to be available in OpenSSH format.</td>
</tr>
<tr>
<td></td>
<td>In such cases, use this download option.</td>
</tr>
</tbody>
</table>
Tip
Want to know how to get an SSH2 public key for the id_rsa/id_dsa key-pair of the tenant key-store?

You can use ssh-keygen for this purpose (to be installed on Windows via Cygwin, for example).

In ssh-keygen, perform the following command:

```
$ ssh-keygen -e -f id_rsa.pub -m RFC4716 > id_rsa.pub_ssh2
```

Example result file:

```plaintext
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "1024-bit RSA, converted by d0231018WDFN33758618A from OpenSS"
AAAAB3NzaC1yc2EAAAADAQABAAAAgQCRB+UOpmnPF9W4cqn6h1z3V5izPFAOTCXSF4cfnw
ZivjhV82AcKq6QwV/SyOxkXxWp/wObjKgTXtiHngdJ0kyQ+66E1eq/yh04N0J0QM3Vzv15
IhL+eLYBtynk1ddFS16kDLgSz1evA9F988wK1z/vpI8DVjbY8HJj1QbE8wOSQ==
---- END SSH2 PUBLIC KEY ----
```

**Download Root Certificate**
A file with the name `<alias>_rootCA.cer` is downloaded.

**Note**
If the key pair is self-signed, this action performs the same function as [Download Certificate](#).

**Download Signing Request**
A file with the name `<alias>.csr` is downloaded.

A file with the name `<alias>.cer` is downloaded.

It contains the base64-encoded certificate.

For keystore entries owned by SAP, a restricted set of options is available. For an SAP key you can, for example, not download any signing request.

**Next Steps**

You can import the downloaded files into other keystores and open them using third-party tools (for example, KeyStore Explorer).
4.3.2.13 Deleting Keystore Entries

Context

This option is not available for keystore entries owned by SAP.

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. On the Current tab, select a keystore entry.
3. Choose the (Actions) icon, then select Delete. Alternatively, you can click the keystore entry alias to open the details, and then choose Delete.

⚠️ Caution

You cannot delete SAP-owned keystore entries.

4.3.2.14 Backing Up Keystore Entries

Back up keystore entries owned by the tenant administrator.

Context

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. On the Current tab, choose Back Up (above the table).

Next Steps

You can view the backed-up entries on the Backup tab.
4.3.2.15 Restoring Backed-Up Keystore Entries

Restore backed-up entries of the keystore (owned by the tenant administrator).

Context

You can restore keystore entries that you have backed up. For example, you can restore entries if you delete them from the active tenant keystore by mistake.

**Note**

Restored keystore entries from the keystore backup replace all existing entries of the tenant administrator. Therefore, be careful using this feature as you can delete by mistake existing keystore entries.

Procedure

1. Choose the **Keystore** tile in the **Manage Security** section.
2. On the **Backup** tab, the backed-up keystore entries are displayed in a table.

   The following attributes are displayed for each entry:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alias</strong></td>
<td>Uniquely identifies the keystore entry. A keystore entry can be either be owned by the tenant administrator or by SAP. In case a keystore entry it is owned by SAP, the lock icon (Maintained by SAP) is shown next to the alias. SAP-owned keystore entries provide a reduced set of functions. For example, you cannot update or delete an SAP-owned keystore entry.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **Type** | Indicates whether the entry is a Certificate (X.509 certificate) or a Key Pair (with public and private key and an X.509 certificate chain).  
  ○ Key Pair  
  Consists of a private key and its X.509 certificate chain.  
  ○ Certificate  
  In many cases this is an X.509 root certificate. |
| **Valid Until** | Indicates the expiration date.  
If keys and certificates have expired, the date is highlighted in red. |
| **Last Modified At** | Indicates the date and time the entry was last modified. |
| **Subject DN** | Indicates the Subject distinguished name (DN) which entails information about the owner of the certificate such as the common name of the certificate owner (CN), the organization (O), or the country or region (C). |
| **Issuer DN** | Indicates the Issuer distinguished name (DN) which entails information about the issuer of the certificate such as common name of the certificate issuer (CN), the organization (O), or the country or region (C). |
| **Actions** | Choose the ![button](Actions) button for a key pair or certificate to perform an action. The available actions depend on the type of the keystore entry and whether it is owned by SAP.  
  ○ Rename  
  ○ Update  
  ○ Download  
  ○ Delete  
  ○ Download Certificate  
  ○ Download Public OpenSSH Key  
  ○ Update Signing Response  
  ○ Download Certificate Chain  
  ○ Download Root Certificate  
  ○ Download Signing Request |

3. Choose Restore (above the table).  
The selected entries are restored in the deployed tenant keystore.

**Related Information**

**Backing Up Keystore Entries** [page 74]  
4.3.2.16 Downloading Backed-Up Keystore Entries

Download the public content of the backup keystore to your local disk.

Context

You can download public keystore entries that you have backed up.

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. Go to the Backup tab, where the backed-up keystore entries are displayed in a table.
3. Choose Download (above the table).
   
   The public part of the backup keystore is downloaded (in a file with the name PublicContentBackupKeystore.jks).

4.3.2.17 Managing the Lifecycle of Keys

To enable secure communication between the tenant and connected remote systems, the system keystore that is deployed on the tenant must contain the up-to-date keys owned by the tenant administrator and SAP.

Context

SAP Cloud Platform Integration comes with a set of features that facilitate the tenant administrator’s task of renewing keys provided by SAP on the tenant.

The following different keystore types enable the tenant administrator to renew keys in an efficient way:

- **System keystore**
  This keystore is located in the system.jks file provided with the tenant and contains all keys that are actively used by the deployed integration flows.

- **New SAP Keys keystore**
  This keystore contains keys already prepared by SAP before a key pair expires.
  As the key expiry date approaches, the tenant administrator can do the following:
  - Download the new SAP keys from the KeyRenewal keystore to share them with the administrators of the connected back ends
  - Activate the keys relevant for the tenant
• SAP Key History keystore
  This keystore contains old (expired) keys.
  If required, the tenant administrator can restore a key from the Key History keystore and use it to replace a key in the system keystore.

During the activation of an SAP key, the system performs three steps, as shown in the following figure.

1. The old key pair (which expires soon) is added by the system to the Key History keystore.
2. The old key pair (in the system.jks keystore) is overwritten by the new key pair from the New SAP Keys keystore.
3. In the New SAP Keys keystore, the new key pair is removed.

**i Note**

To restore an old key pair, the tenant administrator can copy the key pair from the SAP Key History keystore to the New Keys keystore and activate it there.

**Procedure**

**Related Information**

*Activating a New Key Pair Provided by SAP* [page 79]
*Restoring a Key Pair from the Key History* [page 81]
*Renewal of Keys Provided by SAP* [page 184]
4.3.2.17.1 Activating a New Key Pair Provided by SAP

Activate a new key pair provided by SAP in order to replace an old key pair which is supposed to expire soon.

Context

⚠️ Caution

Make sure that you activate a new SAP key pair as part of a dedicated key renewal process that you have agreed on with the administrators of all connected sender and receiver systems.

For more information on the recommended processes, see the separate topics.

ℹ️ Note

Note regarding the case if you don’t renew an SAP key prior to its expiration date:
A system job makes sure that in this case the new key is automatically activated (one day before the expiration date).
This job checks daily if there are new entries in the New SAP Keys keystore and if an SAP key pair entry in the system keystore (system.jks) is due to expire within the next 24 hours. If this is the case, the job checks whether the New SAP Keys keystore contains an entry with the same alias. If this is the case, the key pair entry is activated (through the same sequence of steps as shown in the figure).

Procedure

2. Select the entry and choose Activate.

Related Information

Renewal of Keys Provided by SAP [page 184]
Activating a New SAP Key Pair on the Tenant [page 186]
4.3.2.17.2 Downloading a New Key Pair Provided by SAP

Download a single Key Pair to your local disk.

Context

You can download a key, a single certificate, a complete certificate chain or a root certificate from the New SAP Keys tab.

Note

You can download only public keystore content (no private keys).

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. On the New SAP Keys tab, select a keystore entry.
3. Choose one of the following options for the selected keystore entry.

   Download only supports Key Pair entries in the New SAP Keys tab.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Pair entry</td>
<td>You can download the public key (X.509 format).</td>
</tr>
<tr>
<td></td>
<td>○ Download Certificate</td>
</tr>
<tr>
<td></td>
<td>A file with the name &lt;alias&gt;.cer is downloaded.</td>
</tr>
<tr>
<td></td>
<td>○ Download Certificate Chain</td>
</tr>
<tr>
<td></td>
<td>A file with the name &lt;alias&gt;.p7b is downloaded.</td>
</tr>
<tr>
<td></td>
<td>The file contains the whole certificate chain.</td>
</tr>
<tr>
<td></td>
<td>○ Download Root Certificate</td>
</tr>
<tr>
<td></td>
<td>A file with the name &lt;alias&gt;_rootCA.cer is downloaded.</td>
</tr>
</tbody>
</table>

Next Steps

You can import the downloaded files into other keystores and open them using third-party tools (for example, KeyStore Explorer).
4.3.2.17.3 Restoring a Key Pair from the Key History

Restore a key pair from the SAP Key History.

Context

You use this process to restore an entry from the SAP Key History in the New SAP Keys keystore. In a subsequent step, you can then reactivate the key to make it available in the tenant’s system keystore.

⚠️ Caution

Make sure that you activate a new SAP key pair according to the dedicated key renewal process that you have agreed with the administrators of all connected sender and receiver systems.

For more information about the recommended processes, see the separate topics.

Procedure

1. Choose the SAP Key History tile in the Manage Security Material section.
2. Select the entry and choose Add to New SAP Keys.

   In the subsequent dialog, the old SAP alias name is proposed, but you have the option to overwrite it with a new SAP alias.

→ Tip

   You can specify a new SAP alias if you want to avoid overwriting already active key pairs with this step.

Related Information

Renewal of Keys Provided by SAP [page 184]
Activating a New SAP Key Pair on the Tenant [page 186]
4.3.2.17.4 Downloading a Key Pair from the Key History

Download a key pair from the SAP Key History tab.

Context

You can download a key, a complete certificate, a certificate chain, or a root certificate from the SAP Key History tab.

<table>
<thead>
<tr>
<th>i Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can download only public keystore content (no private keys).</td>
</tr>
</tbody>
</table>

Procedure

1. Choose the Keystore tile in the Manage Security section.
2. On the SAP Key History tab, select a keystore entry.
3. Choose one of the following options for the selected keystore entry.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key Pair</strong> entry</td>
<td>You can download the public key (X.509 format).</td>
</tr>
<tr>
<td></td>
<td>○ Download Certificate</td>
</tr>
<tr>
<td></td>
<td>A file with the name &lt;alias&gt;.cer is downloaded.</td>
</tr>
<tr>
<td></td>
<td>○ Download Certificate Chain</td>
</tr>
<tr>
<td></td>
<td>A file with the name &lt;alias&gt;.p7b is downloaded.</td>
</tr>
<tr>
<td></td>
<td>The file contains the whole certificate chain.</td>
</tr>
<tr>
<td></td>
<td>○ Download Root Certificate</td>
</tr>
<tr>
<td></td>
<td>A file with the name &lt;alias&gt;_rootCA.cer is downloaded.</td>
</tr>
</tbody>
</table>

Next Steps

You can import the downloaded files into other keystores and open them using third-party tools (for example, KeyStore Explorer).

Related Information

Restoring a Key Pair from the Key History [page 81]
4.3.3 Managing Certificate-to-User Mappings

The Manage Security Material area provides an overview of security-related artifacts. It also provides access to all certificate-to-user mappings defined for the tenant.

Remember

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.

Note

Client certificate authentication is configured differently in the Cloud Foundry environment. There is no certificate-to-user mapping, but certificates can be mapped to OAuth clients instead.

See:

You can display all certificate-to-user mappings deployed on the tenant by clicking the Certificate-to-User Mapping tile under Manage Security Material.

Certificate-to-User Mapping Overview

A list of certificate-to-user mappings is displayed in a table. For each artifact, the following attributes are displayed:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name</td>
<td>Name of the user to which the certificate is to be mapped</td>
</tr>
<tr>
<td>Subject DN</td>
<td>Indicates the subject distinguished name (DN) which entails information about the owner of the certificate such as the common name of the certificate owner (CN), the organization (O), or the country or region (C).</td>
</tr>
<tr>
<td>Issuer DN</td>
<td>Indicates the Issuer distinguished name (DN) which entails information about the issuer of the certificate such as common name of the certificate issuer (CN), the organization (O), or the country or region (C).</td>
</tr>
<tr>
<td>Serial Number</td>
<td>Unique serial number of the certificate</td>
</tr>
<tr>
<td>Valid Until</td>
<td>Time when the certificate expires</td>
</tr>
<tr>
<td>Last Modified by</td>
<td>Name of user who has modified the certificate-to-user mapping the last time</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Last Modified on</td>
<td>Date and time when certificate-to-user mapping has been modified the last time</td>
</tr>
</tbody>
</table>

**i Note**

Limit for certificate-to-user mapping: 2 MB (corresponds to about 2000 X.509 certificates)

To sort and filter the content of the table, choose **Table Settings**. On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

The search allows you to filter specific certificate-to-user mappings by providing parts of their name.

**Actions**

To add a new artifact, perform the following steps:

1. Choose **Add**.
2. Specify the **User Name** and (next to the **File** field) click **Browse** and search for the certificate file (.cer file) on your computer.
3. Choose **OK**.

To edit an existing Certificate-to-User Mapping artifact, perform the following steps:

1. Select the artifact in the table and choose **Edit**.
2. You have the options to change the User Name, specify (select) another certificate, or to change both attributes.

You can also delete an artifact.
4.3.4 Managing Access Policies

The Access Policies monitor allows you to show and maintain access policies.

In SAP Cloud Platform Integration, user permissions are granted based on tasks that can be performed on all artifacts and data. Access policies provide a way to additionally protect a subset of artifacts and data.

**Note**
The current scope is the protection of Business Data collected during execution of integration flows. More specifically, access policies guard access to the Message Processing Log Attachments, Trace Data, and Messages in JMS queues. This rule applies regardless of whether this data is accessed via Monitoring UI or via APIs.

Access policies are coupled with custom user roles. Thus, per group of integration flows that are to be accessible only for a specific group of users, you’ve to:

- Create a Custom Role on SAP Cloud Platform for your TMN application (follow the procedure described for Custom roles with applications subscribed from other accounts in Managing Roles).
- You have to be an Administrator for your subaccount to maintain custom roles.
  
  **Note**
  To be used for an access policy, the length of the custom role name is limited to a maximum of 200 characters. Leading and trailing spaces aren’t allowed.
  
  **Note**
  Only users allowed to access business data (having AuthGroup.BusinessExpert or permission esbmessagestorage.read) are affected.
  
  **Note**
  Newly assigned roles take Effect only after login renewal.

- Create an access policy containing references to all integration flows of the group.

If at least one access policy is defined, all access attempts to the data in scope checks whether there’s any restriction for the related integration flow. Access is allowed:

- for all users on integration flow not referenced by any access policy,
- for users having the role associated with the access policy referencing the related integration flow.

How to access the Access Policies.
Go the Manage Security section and select the Access Policies tile. The list of existing access policies is displayed.

**i Note**

To maintain access policies, you need to have the Administrator role.

To create a new access policy, select [Create] and provide:

**Create Access Policy**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role Name</td>
<td>Enter a role name (mandatory). The role name is the name of a custom role that you’ve already created in your subscriber account for the subscribed TMN application.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter a description for the artifact reference (optional).</td>
</tr>
</tbody>
</table>

Select Create.

Refresh the content to see the new access policy. You can filter the list by searching in role names and descriptions, and sort the list by role name.

To Edit or Delete an access policy, choose the corresponding item from the Actions menu of the respective row.

To view the details of an access policy, select it in the list. The details are displays the Artifact References defined. Provide:

**Add Artifact References**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a name for the Artifact References (mandatory). Use a descriptive name for the artifacts affected (for example: German Employees). The role name is the name of a custom role that you’ve already created in your subscriber account for the subscribed TMN application.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter an artifact description (optional), which isn’t necessarily the description of the artifact: if you choose the operator Matches, describe the set of artifacts selected by the regular expression (for example: integration flows processing data for German employees).</td>
</tr>
</tbody>
</table>

**Condition**

<table>
<thead>
<tr>
<th>Artifact Type</th>
<th>Choose Integration Flow or Message Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Choose either the name or ID of the integration flow. For message queues, only the name is allowed or supported.</td>
</tr>
</tbody>
</table>
### 4.3.5 Managing Access Policies in the CF Environment

The access policies monitor allows you to show and maintain access policies in the Cloud Foundry environment.

In SAP Cloud Platform Integration, user permissions are granted based on tasks that can be performed on all artifacts and data. Access policies provide a way to additionally protect a subset of artifacts and data.

The current scope is the protection of Business Data collected during execution of integration flows. More specifically, access policies guard access to the Message Processing Log Attachments and the Trace Data. This rule applies regardless of whether this data is accessed via Monitoring UI or via APIs.

Access policies in the CF environment are coupled with role collections. Thus, per group of integration flows that are to be accessible only for a specific group of users, you’ve to:

1. Create an access policy:

   Go the Manage Security section and select the Access Policies tile. The list of existing access policies is displayed.

   To maintain access policies, you need to have the Administrator role.

To create a new access policy, select + (Create) and provide:
Create Access Policy

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role Name</td>
<td>Enter a role name (mandatory). The role name is the name of a custom role that you’ve already created in your subscriber account.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter a description for the role specified (optional).</td>
</tr>
</tbody>
</table>

Refresh the content to see the new access policy. You can filter the list by searching in role names and descriptions, and sort the list by role name.

To **Edit** or **Delete** an access policy, choose the corresponding item from the **Actions** menu of the respective row.

To add **Artifact References** to a specific access policy, select the access policy from the list and select + (Create). Provide:

Add Artifact References

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter an artifact name (mandatory).</td>
</tr>
<tr>
<td>Description</td>
<td>Enter an artifact description (optional), which isn’t necessarily the description of the artifact: if you choose the operator matches, describe the set of artifacts selected by the regular expression (for example: integration flows processing data for German employees).</td>
</tr>
</tbody>
</table>

**Condition**

<table>
<thead>
<tr>
<th>Artifact Type</th>
<th>Preset as: Integration Flow.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
<td>Choose either the name or the ID of the integration flow.</td>
</tr>
<tr>
<td>Value</td>
<td>Enter the ID or the Name of the integration flow if you choose equals</td>
</tr>
<tr>
<td>Expression</td>
<td>Is shown only if you choose matches. Enter a regular expression for ID or Name.</td>
</tr>
</tbody>
</table>

Select **Create**.

To **Edit** or **Delete** an Artifact Reference, choose the corresponding action button of the respective row.

2. Switch to the SAP Cloud Platform Cockpit

In the corresponding subscriber account, go to the **Security Section** > **Role Collections** > **New Role Collection** enter a role collection name, and save it.

Go then to your Process Integration subscription. In the security section, to **Attributes**, which contains a new role attribute called custom_role that allows to define the role concept in the CF environment.
Create a new role with + (Create) enter the role name, and select the CustomRoleTemplate. You have to specify in the value for the attribute custom_role, a role name which was previously configured in the Access Policy section of the Monitoring view. This is the mechanism how custom_roles and access policies are linked.

Select Next. Now, you assign the role you’ve created to the role collection you have previously created for your user and it’s displayed in the Role by Role Template list.

You then have to return to your subscription in the SAP Cloud Platform Cockpit. Go to the Security section, select Trust Configuration and SAP ID Service to look up your user. Enter your E-Mail address and select Assign role Collection to assign the newly created role collection to your user.

As soon as the new role collection is assigned to your user, you have access to the restricted data (trace data and attachments) of the protected integration flow.

**Note**

Newly assigned roles take effect only after login renewal.

### 4.3.6 Managing JDBC Material

Learn how JDBC Materials supports SAP Cloud Platform Integration service for interacting with your database server.

Use JDBC Data Source to establish connection with databases and JDBC Drivers helps you to upload and install third-party compliant JDBC type-4 drivers to connect with third-party databases.

**Related Information**

- Managing JDBC Data Sources [page 89]
- Configure JDBC Drivers [page 93]

### 4.3.6.1 Managing JDBC Data Sources

The JDBC Data Sources allows you to create and manage a cluster of artifact connections to interact with a database (DB). Each data source contains information on database type, and database-specific configuration parameters.

Create a data source to handle a connection setup to a cloud or on-premise database.

When you select a JDBC Data Source, the details are displayed to the right of the pane. The header area provides the following information on:

- **Configuration Parameters**
  - Displays information corresponding to a specific database such as Database Type, User, Database ID and so on.
• **Logs**
  Displays information recorded during deployment of the data source. If the deployment fails, it displays the error details along with other technical information.

**i Note**
To undeploy a data source, select it from the list and choose **Undeploy**. The system displays a warning and requires you to confirm the task. After confirmation the data source is undeployed and removed from the list.

**Supported Databases**

Databases Supported for Cloud Integration Tenants in Neo Environment

<table>
<thead>
<tr>
<th>Platform</th>
<th>DB Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP Cloud Platform</td>
<td>HANA</td>
</tr>
<tr>
<td>SAP Cloud Platform</td>
<td>ASE</td>
</tr>
<tr>
<td>AWS - Amazon RDS</td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td>AWS - Amazon RDS</td>
<td>Oracle</td>
</tr>
<tr>
<td>AWS</td>
<td>HANA - aaS</td>
</tr>
<tr>
<td>Microsoft Azure</td>
<td>SQL</td>
</tr>
<tr>
<td>Microsoft Azure</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>On-Premise Database</td>
<td>SQL Server</td>
</tr>
</tbody>
</table>

Databases Supported for Cloud Integration Tenants in Cloud Foundry Environment

<table>
<thead>
<tr>
<th>Platform</th>
<th>DB Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS - Amazon RDS</td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td>AWS - Amazon RDS</td>
<td>Oracle</td>
</tr>
<tr>
<td>AWS</td>
<td>HANA - aaS</td>
</tr>
<tr>
<td>Microsoft Azure</td>
<td>SQL</td>
</tr>
<tr>
<td>Microsoft Azure</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>On-Premise Database</td>
<td>SQL Server</td>
</tr>
</tbody>
</table>

**Procedure to Configure the JDBC driver for On-Premise Database**

1. Add and deploy the JDBC Driver.
2. Log in to your Cloud Connector subaccount, to establish link between your tenant and the on-premise database.
3. Enable your cloud application to access a back-end system. For more information, see Configure Access Control.
4. Connect your cloud application to an on-premise system. For more information, see Consuming the Connectivity Service.
If you choose to access HANA database on SAP Cloud Platform, then see Configure a Service Channel for an SAP HANA Database.

After you complete the above configuration, now create a JDBC data source for your on-premise database.

**To Add or Edit a JDBC Data Source**

1. Choose the JDBC Data Sources tab, and then choose Add for creating a new data source.
   
   or
   
   Choose Edit to modify an existing data source.

   **i Note**
   
   You need not redeploy an integration flow after editing the data source.

2. Specify the following attributes:

   **Database-Specific Configuration Parameters**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Define a name for the data source.</td>
</tr>
<tr>
<td></td>
<td><strong>i Note</strong></td>
</tr>
<tr>
<td></td>
<td>You allowed to enter only alphanumeric characters.</td>
</tr>
<tr>
<td></td>
<td>Special characters and white space aren’t supported.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Provide a detailed description of the data source.</td>
</tr>
<tr>
<td><strong>Database Type</strong></td>
<td>Select one of the database types that is supported:</td>
</tr>
<tr>
<td></td>
<td>○ Cloud Platform HANA Database – SAP Managed</td>
</tr>
<tr>
<td></td>
<td>○ Cloud Platform ASE Database – SAP Managed</td>
</tr>
<tr>
<td></td>
<td><strong>i Note</strong></td>
</tr>
<tr>
<td></td>
<td>JDBC drivers that are uploaded to provide connectiv-</td>
</tr>
<tr>
<td></td>
<td>ity to databases managed by third party are shown in</td>
</tr>
<tr>
<td></td>
<td>the dropdown.</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>Enter the username corresponding of the target database.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Enter the password corresponding of the target database.</td>
</tr>
<tr>
<td></td>
<td><strong>i Note</strong></td>
</tr>
<tr>
<td></td>
<td>You must re-enter the same password in the Repeat</td>
</tr>
<tr>
<td></td>
<td>Password field to confirm that the provided password</td>
</tr>
<tr>
<td></td>
<td>is correct.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Database ID</strong></td>
<td>Specify the ID of the target database. (only available for Cloud Platform HANA and ASE databases that are managed by SAP Managed)</td>
</tr>
<tr>
<td><strong>Access Token</strong></td>
<td>Specify the access token generated for this application, by which the user is identified to access the databases. SAP Cloud Platform provides the token to access SAP database. For more information for managing access, see Managing Access to Databases for Other Subaccounts. (only available for Cloud Platform HANA and ASE databases that are managed by SAP Managed)</td>
</tr>
<tr>
<td><strong>JDBC URL</strong></td>
<td>Specify the database connection URL that the JDBC driver uses to connect with the cloud or on-premise database. JDBC URL to be used only for the following databases: ○ Amazon RDS (Microsoft SQL Server and Oracle) ○ HANA-aaS</td>
</tr>
<tr>
<td><strong>Location ID</strong></td>
<td>Identifies the location of this Cloud Connector for a specific subaccount. The location ID must be unique per subaccount and should be an identifier that can be used in a URI. To route requests to a Cloud Connector with a location ID, the location ID must be configured in the respective destinations. If you don’t specify any value for Location ID, the default is used. (for on-premise database only)</td>
</tr>
<tr>
<td><strong>Cloud Connector</strong></td>
<td>Select <strong>Cloud Connector</strong> if you want to create an on-premise database. (for on-premise database only)</td>
</tr>
</tbody>
</table>

**i Note**

Don’t select this option if you intend to connect with a cloud database.

3. Deploy the data source.

**Quick Actions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>☰ (Settings)</td>
<td>Helps to sort and filter the content of the table. You can define how the table entries are to be sorted (in ascending or descending order). You can also filter the table entries for certain attributes.</td>
</tr>
<tr>
<td>☯ (Reload)</td>
<td>Reloads the table content.</td>
</tr>
</tbody>
</table>
4.3.6.2 Configure JDBC Drivers

Learn how to upload and deploy JDBC type-4 compliant third-party drivers on Cloud Integration service.

JDBC connectivity provides a mechanism for integration flows deployed on Cloud Integration tenants in Neo and Cloud Foundry environments to connect with Amazon RDS (SQL Server and Oracle) and HANA-aaS databases hosted on AWS using JDBC drivers. The tenant administrator configures JDBC drivers on your Cloud Integration tenant to enable you to establish connection to a database managed by a third-party vendor. Your database vendor should provide the driver or download it from their official website. We highly recommend that you work with the latest version of the third-party JDBC driver. All uploaded drivers undergo a security validation before being deployed on a Cloud Integration tenant.

**Note**

Cloud Services may include features that permit third-party code or tools to be downloaded/uploaded into Customer’s Cloud Service account by (i) Customer or (ii) by SAP on behalf of and at the request of Customer from a non-SAP service for which Customer has a license, to facilitate the exchange of data or enable or improve other data processing activities. Such third-party code or tools are Customer Data. The code or tools may not be used to connect the Cloud Service or other software/cloud services to a third-party database licensed from SAP under a run-time license for use with specified SAP Applications. Customer must license a full-use license from the applicable database vendor to use such code or tools to replicate data from such database into the Cloud Service.

<table>
<thead>
<tr>
<th>Driver Name</th>
<th>Download From</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft JDBC Driver for SQL Server</td>
<td></td>
</tr>
<tr>
<td>Oracle JDBC Driver</td>
<td></td>
</tr>
</tbody>
</table>

You can create Amazon Relational Database Service (Amazon RDS) based data source that supports the following database engines:

- Microsoft SQL Server
- Oracle

Before consuming the drivers in runtime, you must deploy the uploaded driver as a data source to establish a connection with above mentioned databases. For more information on deploying a data source, see Managing JDBC Data Sources [page 89].

**Deploying JDBC Drivers**

1. Choose the *JDBC Driver* tab, and then choose *Add* for uploading a driver from your file system.
2. Specify the following attributes:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database Type</strong></td>
<td>Select one of the database types that are supported:</td>
</tr>
<tr>
<td></td>
<td>○ Microsoft SQL Server</td>
</tr>
<tr>
<td></td>
<td>○ Oracle</td>
</tr>
<tr>
<td><strong>JDBC Driver File</strong></td>
<td>Browse to choose the driver file from your file system.</td>
</tr>
</tbody>
</table>

***Note***

Only JAR (*jar) file formats are supported.

3. Deploy the driver.

**Quick Actions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Settings)</td>
<td>Helps to sort and filter the content of the table. You can define how the</td>
</tr>
<tr>
<td></td>
<td>table entries are to be sorted (in ascending or descending order). You can</td>
</tr>
<tr>
<td></td>
<td>also filter the table entries for certain attributes.</td>
</tr>
<tr>
<td>(Reload)</td>
<td>Reloade the table content.</td>
</tr>
</tbody>
</table>

4.3.7 Managing Custom Roles in the Cloud Foundry Environment

The user role monitor allows a tenant administrator to manage user roles which then can be used during inbound authorization of an integration flow execution.

***Note***

These instructions are relevant only when you use SAP Cloud Platform Integration in the Cloud Foundry environment.

**Remember**

There are currently certain limitations when working in the Cloud Foundry environment. For more information on the limitations, refer to SAP Note 2752867.
**How to Access the User Roles Monitor**

You open the area with the following action:

In the *Monitor* view, click the *User Roles* tile in the *Manage Security* section. User roles defined by SAP and by the tenant administrator are displayed.

**UI Overview**

A list of user roles are displayed in a table. For each role, the following attributes are displayed:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Display name of the user role</td>
</tr>
</tbody>
</table>
|           | **i Note** When creating a new user role, be aware of the following allowed characters in user role names:  
• a-z  
• A-Z  
• 0-9  
• _ (underscore)  
• . (period) |
| Description | Description of the user role |
| Owner      | The owner of the entry  
• **Tenant Administrator**  
The entry is owned by the tenant administrator on customer side.  
• **SAP**  
The entry is owned by SAP. |
| Actions    | Choose one of the two buttons to perform an action. The available actions are:  
• **Edit role** (.Edit role)  
On the subsequent screen, you can edit the description of the selected user role.  
• **Delete role** (Delete role) |

**i Note**

The technical user role `ESBMessaging.send` is predefined by SAP and cannot be removed or changed.
To sort and filter the content of the table, choose **Table Settings** displayed above the table. On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

You also have the option to refresh the table by clicking **Reload content**.

**Actions**

The following actions are available:

- **Add**
  After clicking the button, a subsequent input box appears where you need to specify the role name and the role description. Only the role description is editable afterwards.

- **Download to JSON**
  A JSON file of the selected user roles is downloaded. You need to import the JSON file in the **Process Integration Runtime** tile to create a service instance in the Cloud Foundry environment that is associated with a user role. Only then, you can call an integration flow endpoint and ensure the processing of the integration flow on the runtime node.
  You can either use a custom role that you created, or use the predefined role **ESBMessaging.send**. The JSON content of the role **ESBMessaging.send** is:

    ```json
    {  
      "roles":[
        "ESBMessaging.send"
      ]
    }
    ```

    Instead of downloading the JSON file, you can also use the button **Copy selected roles in JSON format to the clipboard** above the table. In the next step, you can use the copied format to create, as an example, a new service instance.

**Next Steps**

For a detailed instruction on how to create a service instance and a service key in the Cloud Foundry environment, visit

- , and also

Watch the below video to see a basic inbound authentication end-to-end scenario:
4.3.8 Performing Connectivity Tests

You can test the connectivity to a receiver system.

➔ Remember

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.

To access the tests, select Connectivity Tests tile in Manage Security section of Overview page. Different types of connectivity can be tested by selecting the corresponding tab.

Each tab offers a Request Form and a Send button. The response shows a message plus additional connectivity type specific information.

Related Information

SSH Connectivity Tests [page 99]
SMTP Connectivity Tests [page 104]
TLS Connectivity Tests [page 97]
IMAP Connectivity Tests [page 107]
POP3 Connectivity Tests [page 108]
AMQP Connectivity Tests [page 110]
Cloud Connector Connectivity Tests [page 112]

4.3.8.1 TLS Connectivity Tests

When you have chosen the TLS connection the test tool checks the following:

➔ Remember

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.

- If the receiver (host) is reachable for the tenant
- If the keystore is deployed correctly and contains those keys that are required for the specified authentication method during TLS handshake

To perform the TLS connectivity test you need to specify the following settings:
## TLS Connectivity Test Options

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Enter the host name of the receiver. The host name must not contain any path or schema for example, <code>https://</code>. In particular, you must not enter a URL as the host name.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Enter the port that is to be used for outbound communication. Standard port is <strong>443</strong>.</td>
</tr>
<tr>
<td><strong>Authenticate with Client Certificate (optional)</strong></td>
<td>Choose this option if the client is to be authenticated against the receiver (server) during the TLS handshake (a mutual authentication). If you select this option, you can also specify the key to be used for the test (see below).</td>
</tr>
<tr>
<td><strong>Alias (only if Authenticate with Client Certificate has been selected)</strong></td>
<td>Enter the alias that identifies the relevant key pair for client certificate authentication. In case the receiver requires that during the TLS handshake also the tenant (client) is authenticated against the receiver (server), you can configure the connection test accordingly (selecting the <strong>Authenticate with Client Certificate</strong> option). In that case, provide the <strong>Alias</strong> (to indicate which key is to be used from the tenant keystore). In order for the tenant to be able to authenticate itself as the client against the receiver, a suitable key has to be available in the deployed keystore. With the <strong>Alias</strong> field you can narrow down further the check that a specific key is being used (for example, a key that matches the configuration settings in the corresponding adapter).</td>
</tr>
</tbody>
</table>
| **Validate Server Certificate** | Allows you to validate the server certificate. When you have selected the **Validate Server Certificate** option (which is the default setting), the following checks are executed:  
  - If the server certificate belongs to the server the client connects to  
  - If the certificate is signed by an instance the client trusts  
If it was not successful and there is an error message, you can unselect the **Validate Server Certificate** option. |
If the connectivity test was successful, you get the information about the different checks. The **Server Certificates** are displayed. A download option is available allowing you to save and add them to your trust store.

### 4.3.8.2 SSH Connectivity Tests

#### Context

**→ Remember**

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note [2752867](https://support.sap.com).

When you have selected the SSH connection type, the test tool checks if the SSH outbound connection reaches the associated SFTP server.

Depending on the chosen authentication, the following is checked by the test:

- If the server (host) is reachable for the tenant
- If the configured `known_hosts` file deployed on the tenant contains the certificate of the SSH server.

#### Procedure

1. In the *Manage Security* section, select *Connectivity Tests*.
2. Select *SSH*.
3. Define the attributes for the test.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Enter the host name of the receiver. The host name must <strong>not</strong> contain any path or schema (for example, <code>myHost.org</code>). In particular, you must <strong>not</strong> enter a URL as the host name.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Enter the port that is to be used for outbound communication. Standard port is <strong>22</strong>.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Proxy Type</td>
<td>Select a proxy type from the drop-down list:</td>
</tr>
<tr>
<td></td>
<td>○ Internet</td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td>○ On-Premise</td>
</tr>
<tr>
<td></td>
<td>For more information, see</td>
</tr>
<tr>
<td>Location ID</td>
<td>(Only if On-Premise is selected as Proxy Type)</td>
</tr>
<tr>
<td></td>
<td>To connect to a Cloud Connector instance associated with your account, enter</td>
</tr>
<tr>
<td></td>
<td>the location ID that you’ve defined for this instance, in the destination</td>
</tr>
<tr>
<td></td>
<td>configuration on the cloud side</td>
</tr>
<tr>
<td>Timeout (ms)</td>
<td>Specifies a timeout (in milliseconds) after which the connection to the</td>
</tr>
<tr>
<td></td>
<td>server (host) should be terminated. The default value is 10,000 ms.</td>
</tr>
</tbody>
</table>
Authentication

There are the following options:

- **None**
  SFTP server does not require any authentication of the calling component.

- **Public Key**
  SFTP server authenticates the calling component (tenant) based on a public key. If selected, a user name (specified under User Name) and the key (specified under Private Key Alias) are evaluated by the system to authenticate the tenant against the SFTP server.

- **User Credentials**
  SFTP server authenticates the calling component (tenant) based on the user name and password. To make this configuration setting work, you need to define the user name and password in a User Credential artifact and deploy the artifact on the tenant.

- **Dual**
  SFTP server authenticates the calling component (tenant) with two authentication methods: based on a public key and based on user credentials. At runtime, the system evaluates the values of additional parameters in the following way:

  - For the authentication step based on user credentials: Credentials from the deployed artifact with the name given by the Credential Name parameter are evaluated by the system to authenticate the tenant against the SFTP server.
  
  - For the authentication step based on public key: User name contained in the deployed artifact with name given by the Credential Name parameter and the key identified by the Private Key Alias parameter are evaluated by the system to authenticate the tenant against the SFTP server.

**User Name**

(Only if Public Key or Dual is selected as Authentication)

Enter the ID of the user under which the tenant calls the SFTP server.

**Private Key Alias**

(Only if Public Key or Dual is selected as Authentication)

Alias to identify the private key in the keystore used for the communication with the SFTP server.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Credential Name**        | *(only if User Credentials or Dual is selected as Authentication)*  
Name of User Credentials artifact deployed on the tenant.  
This artifact contains user name and password that are used for authentication at the server. |
| **Host Key Verification**  | *(Only if Public Key, User Credentials, or Dual is selected as Authentication)*  
Verifies the host key.  
  - *Against Tenant*  
    Host key is verified against *known_hosts* artifact deployed on tenant.  
  - *Against Partner Directory*  
    Host key is verified against known hosts defined in Partner Directory. |

**iNote**  
You can also load a known_hosts file from the Partner Directory. To point to the Partner Directory content, you need to set the following property in the integration flow before calling the SFTP receiver adapter:  

```
SAP_FtpPdUri
```

You can use, for example, a Content Modifier for this purpose.  

The value of the property needs apply to the following format:  

```
pd:partnerId:parameterId:Binary
```

**iNote**  
It is recommended to use this option with particular caution when you have selected User Credentials as Authentication because in that case you take the risk that you forward the password to an unverified server.
**Check Directory Access**

Only if **Public Key** or **User Credentials** is selected as **Authentication**)

Select this option if you like to check the access to the target directory. If you leave the **Directory** field empty, the default directory is your home directory.

**Restriction**

You can only specify sub-directories of the home directory.

---

### 4.3.8.3 FTP Connectivity Tests

You can perform FTP connectivity tests to check the settings required by the FTP adapter.

**FTP Connection Test**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Enter the host name of the receiver. The host name must not contain any path or schema (for example, https://). In particular, you must not enter a URL as the host name.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Choose or enter the port that is to be used for outbound communication. Standard port is 21 (FTPES/FTP) or 990 (FTPS) depending on the network protocol.</td>
</tr>
</tbody>
</table>
| **Proxy Type** | Select a proxy type from the drop-down list:  
  - **Internet**  
  - **On-Premise** |
| **Location ID** | Only if **On-Premise** is selected as **Proxy Type**.  
  To connect to an SAP Cloud Connector instance associated with your account, enter the location ID that you’ve defined for this instance, in the destination configuration on the cloud side |
### Attribute

**Encryption**

Select the encryption type:
- **Explicit FTPS** (none)
- **Implicit FTPS**
- **Plain FTP**

If you select **Plain FTP**, there is no valid server certificate.

**Authentication**

Choose which mechanism is to be used to authenticate against the server. Possible values are:
- **None**
  - No authentication is attempted. No credential can be chosen.
- **User Credentials**

**Credential Name**

(Only if **User Credentials** has been chosen for **Authentication**)

Choose the name of a deployed credential to use for authentication.

**Valid Server Certificate**

Allows you to validate the server certificate.

When you’ve selected the **Valid Server Certificate** option (which is the default setting), the following checks are executed:
- If the server certificate belongs to the server the client connects to,
- If the certificate is signed by an instance the client trusts.

If it wasn’t successful and there’s an error message, you can unselect the **Validate Server Certificate** option.

**Check Directory Access**

You can check whether you can access a directory.

**Directory**

Select this option if you like to check the access to the target directory. If you leave the **Directory** field empty, the default directory is your home directory.

### 4.3.8.4 SMTP Connectivity Tests

You can perform SMTP connectivity tests to check the settings required for configuring the receiver mail adapter.

→ **Remember**

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.
### SMTP Connection Test

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Enter the host name of the receiver. The host name must <strong>not</strong> contain any path or schema (for example, <code>https://</code>). In particular, you must <strong>not</strong> enter a URL as the host name.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Choose or enter the port that is to be used for outbound communication. Standard port is <strong>587 (SMTP /STARTTLS)</strong> or <strong>465 (SMTPS)</strong> depending on the network protocol.</td>
</tr>
</tbody>
</table>
| **Proxy Type** | Select a proxy type from the drop-down list:  
  - Internet  
  - On-Premise |
| **Location ID** | Only if **On-Premise** is selected as **Proxy Type**. To connect to an SAP Cloud Connector instance associated with your account, enter the location ID that you’ve defined for this instance, in the destination configuration on the cloud side. |
| **Protection** | Select the protection type:  
  - Off (none)  
  - SMTPS  
  
  **i Note**  
  The secure protocol is only available for the **Proxy Type Internet**.  
  - STARTTLS Mandatory  
  - STARTTLS Optional |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Choose which mechanism is to be used to authenticate against the server. Possible values are:</td>
</tr>
</tbody>
</table>
|                            | • None  
  No authentication is attempted. No credential can be chosen.                                                                            |
|                            | • Encrypted User/Password  
  The user name and password aren’t sent in plain text to the server. This authentication mechanism is secure even without a secure connection. |
|                            | • Plain User/Password  
  The user name and password are sent in plain text. Only use this option together with SSL or TLS, as otherwise an attacker could obtain the password. |
| Credential Name            | (Only if User Credentials has been chosen for Authentication) Choose the name of a deployed credential to use for authentication.            |
| Validate Server Certificate| Allows you to validate the server certificate (only when Protection is not Off ).                                                           |
|                            | When you’ve selected the Validate Server Certificate option (which is the default setting), the following checks are executed:                |
|                            | • If the server certificate belongs to the server the client connects to,                                                                 |
|                            | • If the certificate is signed by an instance the client trusts.                                                                             |
|                            | If it wasn’t successful and there’s an error message, you can unselect the Validate Server Certificate option.                              |
| Check Mail Addresses       | Enter sender and optionally receiver email address.                                                                                         |
| From                       | If you have selected Check Mail Addresses, enter the sender’s email address.                                                                  |
| To                         | If you have selected Check Mail Addresses, enter the receiver’s email address.                                                                |

If the connectivity test was successful, you get the information about the different checks. The server certificates are displayed. A download option is available allowing you to save and add them to your trust store.
# 4.3.8.5 IMAP Connectivity Tests

When you have chosen the IMAP (Internet Message Access Protocol) connection, the test tool checks the following:

> **Remember**

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.

- If the receiver (host) is reachable for the tenant

## IMAP Connectivity Test Options

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Enter the host name of the receiver. The host name must <strong>not</strong> contain any path or schema, for example, <code>https://</code>. In particular, you must <strong>not</strong> enter a URL as the host name.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Enter the port that is to be used for outbound communication. Standard port is <strong>143 (IMAP/STARTTLS)</strong> or <strong>993/IMAPS</strong></td>
</tr>
</tbody>
</table>
| **Proxy Type**    | Select a proxy type from the drop-down list:  
|                   |  - **Internet**  
|                   |  or  
|                   |  - **On-Premise** |
| **Location ID**   | Only if **On-Premise** is selected as **Proxy Type**. To connect to an SAP Cloud Connector instance associated with your account, enter the location ID that you’ve defined for this instance, in the destination configuration on the cloud side |
| **Protection**    | Select the protection type:  
|                   |  - **Off** (none)  
|                   |  - **IMAPS**  
| **iNote**         | The secure protocol is only available for the **Proxy Type Internet**.  
|                   |  - **STARTTLS Mandatory**  
|                   |  - **STARTTLS Optional** |

Operations Guide  
Monitoring  
PUBLIC  
107
### Attribute Description

**Authentication**
Choose the mechanism to authenticate against the server. Possible values are:
- **Encrypted User/Password**
  The user name and password aren’t sent in plain text to the server. This authentication mechanism is secure even without a secure connection.
- **Plain User/Password**
  The user name and password are sent in plain text. Use this option only together with SSL or TLS, as otherwise an attacker could obtain the password.

**Credential Name**
Enter a credential name or select one from the drop-down list.

**Validate Server Certificate**
Allows you to validate the server certificate (only when Protection is not **Off**).

When you’ve selected the Validate Server Certificate option (which is the default setting), the following checks are executed:
- If the server certificate belongs to the server the client connects to,
- If the certificate is signed by an instance the client trusts.

If it wasn’t successful and there’s an error message, you can unselect the Validate Server Certificate option.

- **List Folders**
  or for
- **Check Mailbox Content**

If you check List Folders, you get a list of mail folders.
If you select Check Mailbox Content, the folders are checked and the total number of mails and the number of unread mails is displayed.

**Folder (mandatory)**
If you select Check Mailbox Content, you can specify a folder.

If the connectivity test was successful, you get the information about the different checks. The server certificates are displayed. A download option is available allowing you to save and add them to your trust store.

### 4.3.8.6 POP3 Connectivity Tests

When you have chosen the POP3 (Post-Office Protocol) the test tool checks the following.

> **Remember**

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.
**IMAP Connectivity Test Options**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Enter the host name of the receiver.</td>
</tr>
<tr>
<td></td>
<td>The host name must <strong>not</strong> contain any path or schema for example, <code>https://</code>). In particular, you must <strong>not</strong> enter a URL as the host name.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Enter the port that is to be used for outbound communication.</td>
</tr>
<tr>
<td></td>
<td>Standard port is <strong>110</strong> or <strong>995</strong></td>
</tr>
<tr>
<td><strong>Proxy Type</strong></td>
<td>Select a proxy type from the drop-down list: <em>(● Internet or ● On-Premise)</em></td>
</tr>
<tr>
<td><strong>Location ID</strong></td>
<td>Only if <strong>On-Premise</strong> is selected as <strong>Proxy Type</strong>.</td>
</tr>
<tr>
<td></td>
<td>To connect to an SAP Cloud Connector instance associated with your account, enter the location ID that you’ve defined for this instance, in the destination configuration on the cloud side.</td>
</tr>
<tr>
<td><strong>Protection</strong></td>
<td>Select the protection to be used, from the drop-down list.</td>
</tr>
<tr>
<td><strong>i Note</strong></td>
<td>STARTTLS is not supported for port 995. For port <strong>110</strong>, STARTTLS is supported and checked.</td>
</tr>
<tr>
<td></td>
<td>Select the protection type:</td>
</tr>
<tr>
<td></td>
<td>● <strong>Off</strong> (none)</td>
</tr>
<tr>
<td></td>
<td>● <strong>POP3S</strong></td>
</tr>
<tr>
<td><strong>i Note</strong></td>
<td>The secure protocol is only available for the <strong>Proxy Type Internet</strong>.</td>
</tr>
<tr>
<td></td>
<td>● <strong>STARTTLS Mandatory</strong></td>
</tr>
<tr>
<td></td>
<td>● <strong>STARTTLS Optional</strong></td>
</tr>
</tbody>
</table>
## Attribute | Description
--- | ---
**Authentication** | Choose which mechanism is to be used to authenticate against the server. Possible values are:
- **Encrypted User/Password**
  The user name and password are not sent in plain text to the server. This authentication mechanism is secure even without a secure connection.
- **Plain User/Password**
  The user name and password are sent in plain text. You should only use this option together with SSL or TLS, as otherwise an attacker could obtain the password.

**Credential Name** (mandatory) | Enter a credential name or select one from the drop-down list.

**Validate Server Certificate** | Allows you to validate the server certificate (only when Protection is not Off).

When you have selected the Validate Server Certificate option (which is the default setting), the following checks are executed:
- If the server certificate belongs to the server the client connects to
- If the certificate is signed by an instance the client trusts

If it was not successful and there is an error message, you can unselect the Validate Server Certificate option.

**Check Mailbox Content** | If you check Check Mailbox Content the mailbox is checked and the total number of mails in the inbox is displayed.

If the connectivity test was successful, you get the information about the different checks. The server certificates are displayed. A download option is available allowing you to save and add them to your trust store.

### 4.3.8.7 AMQP Connectivity Tests

If you choose the AMQP (Advanced Message Queuing Protocol, the test tool checks if the connection is successful or not.

1. To perform the AMQP connectivity test, you need to specify the following settings:
### AMQP Connectivity Test Options

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transport Protocol</strong></td>
<td>Select a transport protocol from the drop-down list. You can choose between</td>
</tr>
<tr>
<td></td>
<td>• <strong>TCP</strong> and</td>
</tr>
<tr>
<td></td>
<td>• <strong>WebSocket</strong></td>
</tr>
<tr>
<td></td>
<td>Default value is <strong>TCP</strong></td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Enter the host of the messaging server (mandatory).</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Enter the port of the messaging server (mandatory). Default values are:</td>
</tr>
<tr>
<td></td>
<td>• <strong>5671</strong> for TCP or</td>
</tr>
<tr>
<td></td>
<td>• <strong>443</strong> for WebSocket</td>
</tr>
<tr>
<td><strong>Path</strong></td>
<td>Additional attribute for the WebSocket protocol.</td>
</tr>
<tr>
<td></td>
<td>Access path of the messaging server. Default value is <strong>empty</strong>.</td>
</tr>
<tr>
<td><strong>Proxy Type</strong></td>
<td>Select a proxy type from the drop-down list:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Internet</strong> or</td>
</tr>
<tr>
<td></td>
<td>• <strong>On-Premise</strong></td>
</tr>
<tr>
<td><strong>Location ID</strong></td>
<td>Only if <strong>On-Premise</strong> is selected as <strong>Proxy Type</strong>.</td>
</tr>
<tr>
<td></td>
<td>To connect to an SAP Cloud Connector instance associated with your account,</td>
</tr>
<tr>
<td></td>
<td>enter the location ID that you’ve defined for this instance, in the destina-</td>
</tr>
<tr>
<td></td>
<td>tion configuration on the cloud side</td>
</tr>
<tr>
<td><strong>Connect with TLS</strong></td>
<td>Enable the secure connection via TLS (Transport Layer Security).</td>
</tr>
<tr>
<td></td>
<td>Default value: checkbox selected.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Validate Server Certificate | *Note* Only applies if you’ve selected *Connect with TLS.* Allows you to validate the server certificate. When you’ve selected the Validate Server Certificate option (which is the default setting), the following checks are executed:  
  - If the server certificate belongs to the server the client connects to,  
  - If the certificate is signed by an instance the client trusts  
If it wasn’t successful and there’s an error message, you can unselect the Validate Server Certificate option. |

2. Press *Send*.

If the connectivity test was successful, you get the information about the different checks. The server certificates are displayed. A download option is available allowing you to save and add them to your trust store.

### 4.3.8.8 Cloud Connector Connectivity Tests

When you have chosen the Cloud Connector, the test tool checks whether the Cloud Connector has been configured and can be reached.

> **Remember**

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note [2752867](#).

To use the Cloud Connector, you must have installed and configured it as described in

To perform the connectivity test for the Cloud Connector enter a *Location ID* and choose *Send*.

> **Note**

The location ID identifies a specific Cloud Connector and is unique per subaccount. A cloud connector with an empty location ID can also exist.

If the system could not connect to the cloud connector, check the configuration and whether the location ID exists.
4.4 Managing Stores

The Manage Stores section allows you to manage various temporary data storages on your tenant.

Related Information

Managing Data Stores [page 113]
Managing Variables [page 115]
Managing Message Queues [page 116]
Managing Number Ranges [page 122]

4.4.1 Managing Data Stores

Context

The Manage Stores section provides an overview of storages on the tenant, which are temporarily used to persist data of different kind during message processing.

Procedure

1. To open the Manage Data Stores view, choose the Data Stores tile.
2. On the left pane of Manage Data Stores the name of the data store is displayed. Below the data store name the following information is displayed:
   ○ The name of an integration flow - in case the the data store is used by one integration flow (local data store)
   ○ The string Global - in case the data store is shared across all integration flows deployed on the tenant (global data store)
   Furthermore, the number of entries available in the data store and (if applicable) the number of overdue entries is shown for each data store.
3. On top of the list of data stores, you can access the following functions:
### Table Settings

<table>
<thead>
<tr>
<th>Description</th>
<th>Filter by Name</th>
<th>Sort By</th>
<th>Filter By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allows you to filter for data store names that contain a certain string.</td>
<td></td>
<td>Allows you to sort the messages either in ascending or descending order. You can select one of the following sorting criteria:</td>
<td>Allows you to filter by message using the following sorting criteria:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>○ Name</td>
<td>○ Status: Filters data stores with or without overdue entries (you can select either one or both options).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>○ Visibility</td>
<td>○ Visibility: Filters the data stores by the global or technical name of the integration flow.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>○ Total Number of Entries</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>○ Number of Overdue Entries</td>
<td></td>
</tr>
</tbody>
</table>

### 4. On the right side of the screen, details about the data store selected from the left side are shown.

The following attributes are displayed for each data store entry:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>ID of the data store entry. The data store entry ID can be specified by the integration flow developer when designing the data store Write step (Entry ID parameter) or, in case not specified during integration flow design, a unique ID is generated by the system when the entry is written to the data store at runtime.</td>
</tr>
<tr>
<td>Message ID</td>
<td>ID of the message processing associated with the data store entry. Allows you to navigate to the related message processing log.</td>
</tr>
<tr>
<td>Status</td>
<td>Status of the data store entry. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>○ Waiting: Status after the entry has been written into the data store (and is waiting to be consumed by another integration flow)</td>
</tr>
<tr>
<td></td>
<td>○ Overdue: Status in case the entry has not been consumed in the specified due time</td>
</tr>
<tr>
<td>Due At</td>
<td>Date and time the entry is to be consumed (as configured in the data store Write step, Retention Threshold for Alerting parameter)</td>
</tr>
</tbody>
</table>
### 4.4.2 Managing Variables

The Variables view allows you to monitor variables used in integration flows.

Choose the Variables tile in the Manage Store section. You get an overview of the existing variables, with the following attributes.

<table>
<thead>
<tr>
<th>Table Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The variable name is defined in the integration flow.</td>
</tr>
<tr>
<td>Visibility</td>
<td>A variable can be globally visible across all deployed integration flows of the tenant or be used only by one integration flow.</td>
</tr>
<tr>
<td>Integration Flow</td>
<td>Displays the ID of the integration flow the variable is used in.</td>
</tr>
</tbody>
</table>
### Table Settings

<table>
<thead>
<tr>
<th>Description</th>
<th>Updated At</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shows date and time when the variable content was last updated.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Retain Until</th>
</tr>
</thead>
<tbody>
<tr>
<td>Displays date and time until the variable is still available. The retention time is updated along with any update of the variable.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can download the variable content or delete the variable.</td>
<td></td>
</tr>
</tbody>
</table>

You can filter in the table either by variable name or integration flow.

**Note**

To view or download variables you need the authorization `ESBDataStore.readPayload`.

By clicking on the variable name in the table, you can see its content. If the variable content is not defined as a string value, its content cannot be displayed and a message is shown. You can also download the variable by choosing Download. If you choose to save the variable, the system creates a `.zip` file, containing the header properties file.

**Note**

If you want to delete a variable, check that this variable is no longer in use, as there is no “where used list” available.

### 4.4.3 Managing Message Queues

Certain adapters allow you to store messages in queues. Using the Web UI, you can monitor queues that are active for a tenant.

**Remember**

There are currently certain limitations when working in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.

To open the queue-monitoring application, open the Web UI for your tenant and choose Monitor Message Processing.

If one or more queues are active for your tenant, a Message Queues tile is displayed (under Manage Stores). Click the tile to access more information about the message queues.

**Note**

Message queues are only supported by the following adapter types: AS2 adapter, JMS adapter, XI adapter.
**JMS Resources**

In the information box below the header you can see the status of the current JMS Resource usage: Ok (all resources show status OK), Critical (at least one of the resources with status Critical), Exhausted (at least one of the resources with status Exhausted). Select *Details* to get detailed information about the JMS resources.

### JMS Resource Screen

<table>
<thead>
<tr>
<th>Number of Queues</th>
<th>Number of queues already in use in reference to the maximum number of queues available.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The Number of Queues row illustrates the remaining resources:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Ok</strong> (green): multiple queues remaining.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Critical</strong> (orange): only one or no queue remaining.</td>
</tr>
</tbody>
</table>

**Note**


<table>
<thead>
<tr>
<th>Capacity</th>
<th>Number of MBs already in use in reference to the maximum number of MBs available.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The Capacity row illustrates the remaining resources:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Ok</strong> (green): Informs the user that the total number of MBs available is &gt; 20%.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Critical</strong> (orange): Informs the user that the total number of MBs available is &lt; 20%.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Exhausted</strong> (red): Informs the user that the total number of MBs has reached 95% of the overall available storage.</td>
</tr>
</tbody>
</table>

**Note**

Queue Status
Displays a summary of the individual statuses of the respective queues.

**i Note**
The color changes according to the total queues’ statuses.

- **Ok** (green): all queues display the status Ok.
- **Critical** (orange): min. one queue with status Warning.
- **Exhausted** (red): min. one queue with status Error.

Transactions
Consumers, providers, and an open transaction are always required in order to process a message at runtime. If a message is stored to the JMS queue (JMS Receiver channel), a provider is required; for polling a message from the JMS queue (JMS sender channel), a consumer is required.


Providers
Consumers

Runtime Node
Number of Runtime Nodes in use.

Tenant Management Nodes
Number of Tenant Management Nodes in use.

For more information, see the SAP Community blog: [Cloud Integration - JMS Resources and Size Limits](https://blogs.sap.com/2017/10/04/cloud-integration-jms-resource-and-size-limits-in-cpi-enterprise-edition/).

### Manage Message Queues

All message queues that are active for the tenant are displayed in a table. For each queue, the following information is displayed:

**Message Queue Overview**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Name of JMS queue.</td>
</tr>
<tr>
<td><strong>Entries</strong></td>
<td>Number of messages stored in queue.</td>
</tr>
</tbody>
</table>
**Attribute** | **Description**
--- | ---
Status | Status of the sub-queues.

---

**i Note**

One queue consists of three sub-queues that are responsible for processing the JMS message.

The following status values are possible:

- **Ok**: Informs the user that the usage of all sub-queues is < 80%.
- **Warning**: Informs the user that the usage of at least one sub-queue is between 80 and 95%.
- **Error**: Informs the user that the usage of at least one sub-queue is > 95%.

---

To sort and filter the content of the table, choose **Table Settings** (📸). On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

The search allows you to filter specific queues by providing parts of their name.

Select a queue and choose **Actions** (🪄) to perform the following actions:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retry</td>
<td>Triggers a retry of all messages in the selected queue.</td>
</tr>
</tbody>
</table>
| Status | Displays the status details of the following sub-queues:  
- Processing Queue: Queue where the main processing queue messages are consumed from and sent to  
- Error Queue: Queue in which messages are stored until the next scheduled retry  
- Chunking Queue: Queue in which message chunks of large messages are stored.  

**i Note**

CPI automatically splits large messages into smaller chunks of up to 5 MB to enable processing, otherwise they would not be supported by the underlying messaging system. The large original message is stored in the main processing queue.

| Configure Size | Change the maximum size of the sub-queues. |
**Where-Used**

Displays the integration flows in which a queue is used, and specifies whether the integration flows write to or consume a queue, or both. If you select the link to the integration flow, it opens in read-only mode and you can check the scenario that is using the queue.

During the operation of your scenarios, you may find that messages are piling up in one queue, and you want to get more details so that you can analyze why the messages are not being processed.

**Move**

Moves all messages from this queue to another queue.

**Delete**

Deletes the selected queue and all messages stored in the queue.

**Note**

After performing this action, you need to redeploy the integration flow. After deleting a queue, the integration flow cannot write any more messages into the queue and, therefore, cannot process the messages correctly.

To perform general actions, which affect all queues, choose from the following buttons:

**General Actions**

**Attribute** | **Description**
--- | ---
**Check** | You use this function to show unused and missing queues. The check results show the following:
- Queues that are not used in any of the deployed integration flows.
- Queues that are referenced by integration flows but do not actually exist because they have been deleted by mistake.
You can delete any queues that are not needed, and generate any missing queues by redeploying the integration flow.

**Reload** | Allows you to reload the table.

**Sort** | Allows you to sort the table entries according to the displayed sorting criteria.

**Message Queue Details**

To display more details for a message queue, click on the relevant row in the table.
The message queue details view shows all messages of a queue in a table. The following information is displayed for each message:

**Message Queue Details**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMS Message ID</td>
<td>Allows you to jump directly to the message processing log for the message, where you can find the integration flow name, the time the message was sent, and other message processing details.</td>
</tr>
<tr>
<td>Message ID</td>
<td>Identifies the associated message processing log Click the Message ID to access the associated message processing log.</td>
</tr>
<tr>
<td>Status</td>
<td>The following values are possible:</td>
</tr>
<tr>
<td></td>
<td>● Waiting The message is still waiting to be processed</td>
</tr>
<tr>
<td></td>
<td>● Overdue The message has not yet been processed and the deadline (Due At date) has passed</td>
</tr>
<tr>
<td></td>
<td>● Failed An attempt was made to process the message but an error occurred</td>
</tr>
<tr>
<td></td>
<td>● Blocked The message was involved in multiple node crashes and was therefore not processed The message is not automatically restarted as it was probably this message that caused the node to crash. You can see that the status is Blocked, and also that the Retry At column is empty. Check the message size and the integration flow. The message may be too large or there could be an error in the integration flow. You can then either delete the message or perform a manual retry in the Queue Monitor. Make sure that the problem that caused the crash in the integration flow has been resolved. Maybe the size of the virtual machine (VM) has to be increased.</td>
</tr>
<tr>
<td>Due At</td>
<td>Due date for the message. The due date is calculated as follows: The number of days configured as the Retention Threshold for Alerting in the adapter that the message originates from is added to the date on which the message is created. Once the due date is reached, the status of the message is set to Overdue.</td>
</tr>
<tr>
<td>Created At</td>
<td>Date when the message was stored in the queue.</td>
</tr>
<tr>
<td>Retry Count</td>
<td>Number of retries of the message.</td>
</tr>
<tr>
<td>Next Retry On</td>
<td>Date when next retry is scheduled.</td>
</tr>
<tr>
<td>Retain Until</td>
<td>Date until which the message is retained. Once this date is reached, the message is automatically deleted.</td>
</tr>
</tbody>
</table>

You can perform the following actions for a selected message:

- **Retry**
  Triggers retry of the selected message. To be able to trigger a message retry, you need the following authorization:
Delete
Deletes the selected message. To be able to delete a message, you need the following authorization:
- AuthGroup.Administrator or AuthGroup.IntegrationDeveloper

Download
Downloads the message with attachments as a zip file. To be able to download a message, you need the following authorization:
- AuthGroup.BusinessExpert

The zip file has four entries with the following names:
- attachments: Folder that contains message attachments (as .txt files)
- body: Contains the message body.
- properties.prop: Contains the exchange properties.
- headers.prop: Contains the headers.

**Note**
If a JMS message has several attachments, a zip entry is generated for each attachment. The names of the attachment entries have the following form: attachment_<File Name | Content ID | UUID>
- <File Name | Content ID | UUID> means that either the file name, the content ID header, or a UUID is used as the name component, depending on whether the value for the file name or the content ID is available.
- If there is no attachment, the zip file does not contain an attachment entry.
- The naming convention for the entry with the message body is body_<Content ID>, where _<Content ID> is only included if there is a content ID header for the body.
- The entries exchange.properties and header.properties are only included if the JMS message has properties or headers.

**Related Information**


**4.4.4 Managing Number Ranges**

The topic provides an overview of number ranges related artifacts.

You choose the **Number Ranges** tile in the **Manage Stores** area to view the artifacts with the corresponding status.
Number Ranges Overview

While sending out a document in case of EDI processing, a unique interchange number must be added to each document. In order to add such an interchange number you can use the Number Range Object. In case of EDIFACT the outgoing EDIFACT messages should have Interchange Control Reference with length 1 to 9 digits, with minimum length being 1 digit and maximum length ranging to 9 digits.

Visit the blog, to understand how to consume Number Ranges in EDI message processing.

A list of number ranges is displayed in a table. For each artifact, the following attributes are displayed:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Displays name of the artifact</td>
</tr>
<tr>
<td>Minimum Value</td>
<td>Minimum value of the artifact should be greater than or equal to 0.</td>
</tr>
<tr>
<td>Maximum Value</td>
<td>Maximum value of the artifact should be less than 15 digit.</td>
</tr>
<tr>
<td>Next Value</td>
<td>Displays a value that can be used the next time you invoke this artifact.</td>
</tr>
<tr>
<td>Field Length</td>
<td>Displays the number of digits for the current value. If the value of  maximum value attribute is 100 then the field length should be greater than 2, if the value of  maximum value attribute is 99999 then the field length should be greater than 4 and so on. If the value of this attribute is 4 and the value of current value attribute is 7, then the number range is reflected as 0007 and not just 7. This refers to the concept of padding '0's to the value. If the value of this attribute is 0 then the value of current value attribute is flashed as it is with no padding on the number range. The maximum value allowed for this attribute is 99.</td>
</tr>
<tr>
<td>Rotate</td>
<td>If this attribute is set and the number range reaches specified maximum value, then the current value resets to specified minimum value.</td>
</tr>
<tr>
<td>Deployed By</td>
<td>Displays the user id of the user who deployed the artifact.</td>
</tr>
<tr>
<td>Deployed On</td>
<td>Displays the time when the artifact was deployed.</td>
</tr>
</tbody>
</table>

To sort and filter the content of the table, choose Table Settings ( ). On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.
Actions

- To create or deploy a new artifact, choose \textit{Add}.
- To edit and redeploy an existing artifact, select the artifact in the table and choose \textit{Edit}.
- To undeploy an artifact, select the artifact in the table and choose \textit{Undeploy}.

4.5 Accessing Logs

The \textit{Access Logs} section allows you to monitor audit logs (resulting from system changes) and to analyze errors that occurred during inbound HTTP processing (and documented in system log files).

Related Information

Monitoring Audit Log [page 124]
Monitoring System Log Files [page 125]

4.5.1 Monitoring Audit Log

The audit log contains information on system changes. These events can be for example the deployment of an integration flow as well as a configuration change.

\rightarrow \textbf{Remember}

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.

You view the audit log by clicking the tile \textit{Audit Log} in the \textit{Manage Security} area.

\rightarrow \textbf{i Note}

To view the content of the log, you need the authorization \textit{AuditLog\_Read}.

You can control the display of the messages by adjusting the filter setting \textit{Time Range}.

You can choose from the following preset time ranges:

- All
- Past Hour
- Past 24 Hours
- Past Week
- Custom
The audit log data retention time in the database is 30 days.

You can retrieve the following information from the audit log list and filter the entries by Object Name, User or Source.

### Audit log List

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Displays the time of the event.</td>
</tr>
<tr>
<td>Action</td>
<td>Displays the action performed on the system such as Create or Delete.</td>
</tr>
<tr>
<td>Object Type</td>
<td>Displays the object type such as IntegrationFlow on which the action was performed on.</td>
</tr>
<tr>
<td>Object Name</td>
<td>Displays the object name such as the integration flow name.</td>
</tr>
<tr>
<td>User</td>
<td>Displays the user who triggered the action.</td>
</tr>
<tr>
<td>Source</td>
<td>Displays the IP address of the source that issued the action.</td>
</tr>
</tbody>
</table>

If an SAP user triggers the changes, User and Source are displayed as SAP in the audit log list.

You can also sort the audit log list by Time, Action, Object Type or Object Name.

The audit log retrieves a maximum of 1000 entries from the database. If you have more than 1000 entries in the selected time range, you will be prompted to adjust your filter settings accordingly.

### 4.5.2 Monitoring System Log Files

This section contains information on system log files. These log files can be either HTTP access files or Cloud Platform default trace files.

This component or some of its features might not be available in the Cloud Foundry environment. For more information on the limitations, see SAP Note 2752867.

You view the system log files by clicking System Log Files in the Access Logs area.

You can retrieve the following information from the system log file list:
Log Files

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Displays the log file name</td>
</tr>
<tr>
<td>Log Type</td>
<td>Displays 2 different log types the CP default trace or the HTTP access log</td>
</tr>
<tr>
<td>Updated At</td>
<td>Displays the date of the last update</td>
</tr>
<tr>
<td>Size</td>
<td>Displays the file size</td>
</tr>
<tr>
<td>Actions</td>
<td>You can either download the file by selecting 🔗 or get the URL by selecting 🔗</td>
</tr>
</tbody>
</table>

You can filter the log files by names and sort the list either by Name, Updated At, or file Size. You can either download a specific file or get the file URL. The file URL allows you to send it per mail to allow further analysis on another computer for example. In both cases, the system provides a packed log file (.zip file).

i Note
The log file retention time is 7 days.

If you select Collections, you get the most recent log files for each runtime node. You can either download the collection or get the URL and in both cases the system provides packed log files (.zip files).

4.6 Managing Locks

This section allows you to display and manage lock entries that are created (in the in-progress repository) to avoid the same message being processed several times in parallel (for example, by different runtime nodes).

→ Tip
Example:
Several runtime nodes try to read a file from an SFTP server (through SFTP sender channels).
To prevent double processing, a lock entry is written to the in-progress repository each time a file is processed by a runtime node. As long as this lock entry exists, no other component can access the file. After message processing, the lock is removed by the runtime.

In certain situations (for example, a runtime node crashes because of an out-of-memory error), the message is retried after the node is restarted until the expiration time is reached. In this case, lock entries could remain in the in-progress repository and block subsequent message processing. You can use the Manage Locks view to analyze the situation and manually delete lock entries, if required, to reprocess the message.
If you choose the Message Locks tile under Manage Locks tile in the Monitor application, a list of locks is displayed. The following information is shown for each lock entry:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Component that wrote the lock entry</td>
</tr>
<tr>
<td>Source</td>
<td>Component that caused the lock</td>
</tr>
<tr>
<td>Entry</td>
<td>Content of the lock entry</td>
</tr>
<tr>
<td>Created at</td>
<td>Time when the lock was created</td>
</tr>
<tr>
<td>Expires at</td>
<td>Time when lock entry expires</td>
</tr>
</tbody>
</table>

AS2, JMS, Mail, SFTP, and XI adapters can write locks.

Example for SFTP: user_sftp@ld1234.mycompany.corp

Example for SFTP: directory1/dir2/test.xml

If the lock relates to an SFTP connection, the Entry field shows a file name (of the file that the SFTP adapter tries to read from the SFTP server).

This parameter provides a link to the message in the Managing Message Queues monitor. To access the associated JMS Message, click ID. The Queue Monitor opens and loads the JMS Message.

This is only possible for JMS locks.

You can also search for table entries (search field).

To sort and filter the content of the table, choose Table Settings ( ). On the subsequent screen, you can define how the table entries are to be sorted (by specifying an attribute and whether the entries are to be sorted for that attribute in ascending or descending order). You can also filter the table entries for certain attributes.

To remove the lock entry and retriger message processing, select the entry and choose Release.

⚠️ Caution

Before releasing a lock entry, make sure you do a careful problem analysis. In particular, make sure that you have understood how the lock entry in question relates to the actual problem you are trying to solve. Careless usage of the release lock function may lead to data inconsistencies or other serious problems.

For example, in the case of an SFTP connection the lock entry is a file name (including the file path). In this case, check whether the problem is related to the file that is to be processed through SFTP.

Another indicator is the duration of the lock (time that has passed since the time specified under Created at).
4.7 First Steps in Analyzing Runtime Errors

When an error occurs at runtime, the SAP Cloud Platform Integration Web UI (Operations view) provides a set of features that help you to analyze such errors.

Checking the Message Processing Log

Find out if message processing has failed for dedicated integration flows.

To do that, open the SAP Cloud Platform Integration Web UI (Operations view) and select a tile under Monitor Message Processing.

Check for messages in status Failed.

On clicking a tile, a dialog opens where you can select for individual integration flows. When you click an integration flow for which message processing failed, a screen with more details opens. For failed messages, an error message is displayed which might help already to analyze the issue.
In case the information is not sufficient and you might like to investigate in more detail where during message processing an error occurred, you can run the integration flow again with another log level. To do that, in the SAP Cloud Platform Integration Web UI (Operations view) click a tile under Manage Integration Content and select the integration flow you like to investigate.

Click tab Log Configuration and for Log Level select Trace.

Retrigger message processing through this integration flow again (for example, using an HTTP client to call it, in case the sender adapter allows for this option). Very likely, message processing fails again.

You can now again go to the Monitor Message Processing section and select the integration flow. To display more details, select the Logs tab and click the Trace link.
On the next screen, the integration flow model is shown. The integration flow step where an error occurred might already have been marked with a red envelope icon.

If you like to investigate the situation in more detail, you can click individual integration flow steps. If you do that, the actual content of the message payload, the message header and the Exchange property values are displayed for the selected step.

More information:

Monitoring Message Processing [page 25]
Checking the Default Trace

Another option to analyze runtime errors is to check the default trace. To do that, using the SAP Cloud Platform Integration Web UI (Operations view) click the System Log Files tile. Check out the latest files starting with ljs_trace_ (which you can also download to your computer). You notice that there are multiple entries with a similar same time stamp (Updated At). Note that 1 log is written for each runtime node started on the tenant.

More information:
Monitoring System Log Files [page 125]
Cloud Integration – System Log Download (SAP Community blog)

For a detailed instruction how to analyze the default trace in the context of inbound connection issues, check out the following SAP Community blog: Cloud Integration – How to Setup Secure HTTP Inbound Connection with Client Certificates.
Security Artifact Renewal

Security artifacts like certificates or passwords (for example) are subject to a specific lifecycle, in other words, they expire in certain time periods. To make sure that the operation of a scenario (using security artifacts) can be continued without any downtime, the process to renew security artifacts has to be performed in a coordinated way by the administrators of the involved components.

For the different use cases specific security artifact renewal processes have been defined.

Note

Note the following with regard to terminology:

- The terms client and server are preferably used in the context of the certificate-based authentication option for HTTPS-based communication (transport level security). The background of this is that in order to set up a mutual authentication (that comes with this option), certificates for both roles, client and server, are required. When a message is sent from a sender (which has the role of a client) to a receiver (which has the role of a server), authentication steps are executed both to check if the server is a trusted partner and if the client is allowed to call the server.

- In the context of message level security, the terms sender and receiver are preferably used in order to simplify things. These use cases typically require the following kinds of certificates or keys:
  - Keys owned by a sender to either encrypt or sign the content of a message
  - Keys owned by a receiver to either verify or decrypt the content of a message

Related Information

Basic Security Artifact Renewal Processes [page 132]
Renewal of Keys Provided by SAP [page 184]

5.1 Basic Security Artifact Renewal Processes

Note

These topics describe separate processes for renewing the various keys that are involved in an integration scenario using SAP Cloud Platform Integration. The topics cover situations where a key is owned either by the tenant administrator or by the administrator of the sender/receiver system connected to the tenant.

To keep things simple, the individual topics describe idealized situations where a key pair is used in a single step or only in one communication channel. In real-life situations, however, a key pair is typically used in several integration flows, integration flow steps, and communication channels. The tenant administrator needs to know all the steps and channels where the key pair is used to be able to correctly define the key renewal process.
### 5.1.1 Use Cases

The following tables provides a list of all use cases and links to the corresponding renewal procedures.

## Transport Level Security Use Cases

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Authentication Option</th>
<th>Direction</th>
<th>Renewal Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS</td>
<td>Certificate-Based</td>
<td>Outbound</td>
<td>Renewal of Tenant Client Root Certificate (CA) [137]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Renewal of the Tenant Client Certificate [139]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Renewal of Receiver Back-End Server Certificate [141] (also applicable in case a SuccessFactors receiver channel is used)</td>
</tr>
<tr>
<td>Inbound</td>
<td></td>
<td></td>
<td>Renewal of Load Balancer Server Certificate [142]</td>
</tr>
<tr>
<td></td>
<td>Basic</td>
<td>Outbound</td>
<td>Renewal of User and Password [145] (also applicable in case a SuccessFactors receiver channel is used)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Renewal of Password Only [146] (also applicable in case a SuccessFactors receiver channel is used)</td>
</tr>
<tr>
<td>Inbound</td>
<td></td>
<td></td>
<td>Renewal of User and Password [147]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Renewal of Password Only [147]</td>
</tr>
<tr>
<td>SFTP</td>
<td>Certificate-Based</td>
<td>Tenant pulls data from SFTP server</td>
<td>Renewal of the SFTP Server Key [148]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tennant pushes data to SFTP server</td>
<td>Renewal of the SFTP Client Key (on Tenant) [149]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Renewal of User on SFTP Server [149]</td>
</tr>
</tbody>
</table>
## Message Level Security Use Cases

### Security Artifact Renewal Use Cases (Message Level Security)

<table>
<thead>
<tr>
<th>Standard</th>
<th>Protection Method</th>
<th>Renewal Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS/PKCS#7</td>
<td>Signer</td>
<td>Renewal of Keys for CMS/PKCS#7 Signer - Outbound [page 151] (key pair renewed on tenant)</td>
</tr>
<tr>
<td></td>
<td>(Tenant signs outbound message)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verifier</td>
<td>Renewal of Keys for CMS/PKCS#7 Verifier - Inbound [page 154] (key pair renewed by sender)</td>
</tr>
<tr>
<td></td>
<td>(Tenant verifies inbound message)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Encryptor</td>
<td>Renewal of Keys for CMS/PKCS#7 Encryptor - Outbound [page 156] (key pair renewed by receiver)</td>
</tr>
<tr>
<td></td>
<td>(Tenant encrypts outbound message)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Decryptor</td>
<td>Renewal of Keys for CMS/PKCS#7 Decryptor - Inbound [page 159] (key pair renewed on tenant)</td>
</tr>
<tr>
<td></td>
<td>(Tenant decrypts outbound message)</td>
<td></td>
</tr>
<tr>
<td>OpenPGP</td>
<td>Encryption key</td>
<td>Renewal of OpenPGP Encryption Key - Outbound [page 161] (encryption key renewed by receiver)</td>
</tr>
<tr>
<td></td>
<td>(Tenant encrypts outbound message)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Encryption key</td>
<td>Renewal of OpenPGP Encryption Key - Inbound [page 164] (encryption key renewed on tenant)</td>
</tr>
<tr>
<td></td>
<td>(Tenant decrypts inbound message)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Signer key</td>
<td>Renewal of OpenPGP Signer Key - Outbound [page 165] (signer key renewed on tenant)</td>
</tr>
<tr>
<td></td>
<td>(Tenant decrypts outbound message)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Signer key</td>
<td>Renewal of OpenPGP Signer Key - Inbound [page 167] (signer key renewed by sender)</td>
</tr>
<tr>
<td></td>
<td>(Tenant verifies inbound message)</td>
<td></td>
</tr>
<tr>
<td>XML Digital Signature</td>
<td>Signer</td>
<td>Renewal of Keys for XML Digital Signature Signer - Outbound [page 168]</td>
</tr>
<tr>
<td></td>
<td>(Tenant signs outbound message)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verifier</td>
<td>Renewal of Keys for XML Digital Signature Verifier - Inbound [page 169]</td>
</tr>
<tr>
<td></td>
<td>(Tenant verifies inbound message)</td>
<td></td>
</tr>
<tr>
<td>WS-Security</td>
<td>Signer</td>
<td>Security Artifact Renewal for WS-Security (Tenant Signs Outbound Request) [page 179]</td>
</tr>
<tr>
<td></td>
<td>(Tenant signs outbound request message)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Signer</td>
<td>Security Artifact Renewal for WS-Security (Tenant Signs Inbound Response) [page 175]</td>
</tr>
<tr>
<td></td>
<td>(Tenant signs response message (to an inbound request))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verifier</td>
<td>Security Artifact Renewal for WS-Security (Tenant Verifies Inbound Request) [page 173]</td>
</tr>
<tr>
<td></td>
<td>(Tenant verifies inbound request message)</td>
<td></td>
</tr>
</tbody>
</table>
5.1.2 Involved Roles

The security artifact renewal process requires that different persons perform a sequence of steps in a coordinated way on each side of the communication. The exact sequence depends on the kind of security material which is renewed and on the use case.

Roles in the Security Artifact Renewal Process

<table>
<thead>
<tr>
<th>Role</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender/receiver administrator (at customer side)</td>
<td>Updates the security artifacts owned by the sender/receiver back-end system (for example, the keystore).</td>
</tr>
<tr>
<td>Integration developer</td>
<td>Updates the integration flow in certain use cases.</td>
</tr>
<tr>
<td></td>
<td>It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.</td>
</tr>
<tr>
<td>Tenant administrator</td>
<td>Updates the security artifacts of the tenant (relevant for outbound communication).</td>
</tr>
<tr>
<td></td>
<td>It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.</td>
</tr>
<tr>
<td>Role</td>
<td>Tasks</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>SAP administrator</td>
<td>SAP administrator -Performs further settings at SAP, for example, the update of the security artifacts of the load balancer (relevant for inbound communication).</td>
</tr>
</tbody>
</table>

5.1.3 Security Artifact Renewal for Transport Level Security

5.1.3.1 Security Artifact Renewal for HTTPS-Based Communication

Using HTTPS, you can specify two different authentication options: certificate-based authentication and basic authentication. These options imply a different set of security artifacts for which specific renewal processes exist.

Certificate-based authentication (through HTTPS) involves the usage of X.509 SSL certificates both at client and server side. These certificates expire at a specified point in time. To ensure operation of scenarios using this communication type without any downtime requires the coordinated renewal of certificates both at client and server side.

Basic authentication uses credentials to allow the identification of trusted communication partners. Credentials are composed of user and password and, in case the SuccessFactors connector is involved, additional attributes. In addition to credentials, basic authentication also uses a one-way SSL connection which requires server certificates. Therefore, security artifact update involves both the update of credentials and of the involved certificates.

5.1.3.1.1 Certificate-Based Authentication (Outbound)

Certificate-based outbound authentication involves the usage of X.509 SSL certificates both at client and server side. For outbound communication, the keystore of the tenant is involved.

Related Information

Renewal of the Tenant Client Certificate [page 139]
Renewal of Receiver Back-End Server Certificate [page 141]
5.1.3.1.1.1 Renewal of Tenant Client Root Certificate (CA)

In this use case, the tenant client certificate is exchanged by a new one signed from a different certification authority (CA).

The following figure illustrates the communication path that is relevant for this use case.

![Diagram](image)

**i Note**

It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

**Receiver Accepts Different Certificates at the same Time**

Certificate renewal has to be performed in the following sequence:

1. Tenant administrator: Creates new certificate with a new key pair and gets the certificate signed by another CA than the old one.
2. Tenant administrator: Provides receiver administrator with the new certificate and root certificate (the latter one because the CA had changed).
3. Receiver administrator: Configures receiver (server) that way that the old and the new client certificate are accepted by the server.
   Because the receiver administrator also has received a new root certificate, this root certificate also has to be imported into the server keystore.
4. Receiver administrator: Informs the tenant administrator that the receiver system (server) now accepts both the old and the new client certificate.
5. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate - keeping the old alias in the tenant client keystore.
   This is done by importing the new signed certificate into the tenant client keystore.
6. Tenant administrator: Informs the integration developer that the integration flow can be restarted.
7. Integration developer: Restarts the integration flow which sends data via HTTPS to the receiver system.
This is necessary because the SSL socket caches the keystore for 24 hours. If your tenant cluster contains multiple runtime nodes, make sure that you restart your integration flow on all nodes. If you start the integration flow on only one runtime node, message processing might fail on the other nodes.

8. Tenant administrator: Informs the receiver administrator that a new client certificate (signed by another CA than the old one) is now used.
9. Receiver administrator: Removes the old client certificate and also the old root certificate (assumed that it is not longer used in any other communication).

Let us assume, the customer landscape is composed as described under Connecting a Customer System to Cloud Integration, section Technical Landscape for On Premise-On Demand Integration. In that case, SAP Web Dispatcher is used to receive incoming calls from the SAP Cloud. SAP Web Dispatcher (as reverse proxy) is the entry point for HTTPS requests into the customer system landscape. The configuration of the receiver (server) as indicated in step 2 in the list above comprises the following tasks for that example case:

- Make sure that the reverse proxy trusts the new CA. A restart is required to finalize the related configuration steps.
- Map the new certificate in AS ABAP back-end for authentication purpose
- Edit the new CA in Web Dispatcher farm. This step is performed by SAP IT.
- Upload the new CA in workcenter under Edit Certificate Trust List.
- Update the communication arrangements credentials such way that the new certificate is mapped to the inbound technical user.

**Receiver does not Accept Different Certificates at the same Time**

Certificate renewal has to be performed in the following sequence:

1. Tenant administrator: Creates new certificate with a new key pair and gets the certificate signed by another CA than the old one.
2. Tenant administrator and Receiver administrator: Agree on a downtime window.
3. At start of the downtime window, the tenant administrator informs the integration developer that the integration flow which uses the client certificate for outbound HTTPS communication has to be stopped.
4. Integration developer: Stops the integration flow.
5. Integration developer: Informs the tenant administrator that the integration flow has been stopped.
6. Tenant administrator: Exchanges key pair/certificate in the keystore keeping the old alias. This is done by importing the new signed certificate into the tenant client keystore.
7. Tenant administrator: Provides receiver administrator with the new certificate and the root (CA) certificate. The tenant administrator informs the receiver administrator that the HTTPS client has been stopped.
8. Receiver administrator: Exchanges the certificate and imports the new root (CA) certificate into the truststore.
9. Receiver administrator: Informs the tenant administrator that the certificate has been exchanged.
10. Tenant administrator: Informs the integration developer that the integration flow can be restarted.
11. Integration developer: Restarts the integration flow which sends data via HTTPS to the receiver system. This is necessary because the SSL socket caches the keystore for 24 hours. If your tenant cluster contains multiple runtime nodes, make sure that you restart your integration flow on all nodes. If you start the integration flow on only one runtime node, message processing might fail on the other nodes.
5.1.3.1.1.2 Renewal of the Tenant Client Certificate

In this use case, the tenant client certificate has to be renewed. In the renewal process, the tenant administrator (managing the tenant cluster) and the integration developer (managing the integration flow deployed on the tenant) collaborate with the administrator of the receiver back-end system.

The following figure illustrates the communication path that is relevant for this use case.

Receiver Accepts Different Certificates at the same Time

Certificate renewal has to be performed in the following sequence:

1. Tenant administrator: Creates new certificate with a new key pair and gets the certificate signed by a CA.
2. Tenant administrator: Provides receiver administrator with the new certificate and root certificate (if the CA had changed).
3. Receiver administrator: Configures receiver (server) that way that the old and the new client certificate are accepted by the server. In case the receiver administrator also has received a new root certificate, this root certificate also has to be imported into the server keystore.

4. Receiver administrator: Informs the tenant administrator that the receiver system (server) now accepts both the old and the new client certificate.

5. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate - keeping the old alias in the tenant client keystore. This is done by importing the new signed certificate into the tenant client keystore.

6. Tenant administrator: Informs the integration developer that the integration flow can be restarted.

7. Integration developer: Restarts the integration flow which sends data via HTTPS to the receiver system. This is necessary because the SSL socket caches the keystore for 24 hours. If your tenant cluster contains multiple runtime nodes, make sure that you restart your integration flow on all nodes. If you start the integration flow on only one runtime node, message processing might fail on the other nodes.

8. Tenant administrator: Informs the receiver administrator that the new client certificate is now used.

9. Receiver administrator: Removes the old client certificate and (if required) also the old root certificate (assumed that it is not longer used in any other communication).

### Receiver does not Accept Different Certificates at the same Time

Certificate renewal has to be performed in the following sequence:

1. Tenant administrator: Creates new certificate with a new key pair and gets the certificate signed by a CA.

2. Tenant administrator and Receiver administrator: Agree on a downtime window.

3. At start of the downtime window, the tenant administrator informs the integration developer that the integration flow which uses the client certificate for outbound HTTPS communication has to be stopped.

4. Integration developer: Stops the integration flow.

5. Integration developer: Informs the tenant administrator that the integration flow has been stopped.

6. Tenant administrator: Exchanges key pair/certificate in the keystore keeping the old alias. This is done by importing the new signed certificate into the tenant client keystore.

7. Tenant administrator: Provides receiver administrator with the new certificate and the root (CA) certificate if the latter has been changed. The tenant administrator informs the receiver administrator that the HTTPS client has been stopped.

8. Receiver administrator: Exchanges the certificate and imports the new root (CA) certificate into the truststore (in case a new root certificate has been received).

9. Receiver administrator: Informs the tenant administrator that the certificate has been exchanged.

10. Tenant administrator: Informs the integration developer that the integration flow can be restarted.

11. Integration developer: Restarts the integration flow which sends data via HTTPS to the receiver system. This is necessary because the SSL socket caches the keystore for 24 hours. If your tenant cluster contains multiple runtime nodes, make sure that you restart your integration flow on all nodes. If you start the integration flow on only one runtime node, message processing might fail on the other nodes.

---

**Note**

It is not expected that this case occurs very often.
5.1.3.1.1.3 Renewal of Receiver Back-End Server Certificate

In this use case, the server certificate (of the receiver) has to be renewed.

The following figure illustrates the communication path that is relevant for this use case.

1. **Receiver administrator**: Creates key pair/certificate for the receiver (server) and uses a different CA certificate to sign the server certificate.
2. **Receiver administrator**: Provides the tenant administrator with the server root certificate (of the CA).
3. **Tenant administrator**: Imports the root certificate into the tenant client keystore (of the tenant).
4. Tenant administrator: Restarts all integration flows which are sending via HTTPS data to the receiver system. This is required because the SSL socket caches the keystore for 24 hours.

5. Tenant administrator: Informs receiver administrator that root certificate has been added.

6. Receiver administrator: Exchanges the key pair/certificate in the receiver system.

7. Receiver administrator: Informs the tenant administrator that the old root certificate can be removed.

8. Tenant administrator: Deletes the old root certificate from the tenant client keystore.

Related Information

Involved Roles [page 135]

5.1.3.1.2 Certificate-Based Authentication (Inbound)

Certificate-based outbound authentication involves the usage of X.509 SSL certificates both at client and server side.

5.1.3.1.2.1 Renewal of Load Balancer Server Certificate

In this use case, the load balancer server certificate at SAP has to be renewed. In the renewal process, the load balancer administrator (at SAP) and the sender back-end administrator (at the customer side) collaborate with each other.

The following figure illustrates the communication path that is relevant for this use case.

\[\text{Back-End (Sender)} \rightarrow \text{Load Balancer} \rightarrow \text{SAP} \rightarrow \text{Tenant}\]

\[\text{Keystore} \rightarrow \text{Load balancer server root certificate} \rightarrow \text{Keystore} \rightarrow \text{Load balancer server certificate} \rightarrow \text{System Database}\]

\[\text{i Note}\]

It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and...
Certificate renewal has to be performed in the following sequence.

1. Load balancer administrator: Creates new key pair/certificate with new CA root certificate.
2. Load balancer administrator: Informs tenant administrator that virtual server certificate will be exchanged at a certain point in time and forwards new root (CA) certificate.
3. Tenant administrator: Informs sender administrator and forwards the new root certificate to the sender.
4. Sender administrator: Adds the new root certificate to the truststore of the sender back-end (HTTPS client).
5. Load balancer administrator: Exchanges the load balancer virtual server key pair/certificate at the specified point in time.
6. Sender administrator: Can now remove the old root certificate form the truststore of the sender back-end (HTTPS client) after the specified point in time has passed.

**Related Information**

Involved Roles [page 135]

### 5.1.3.1.2.2 Renewal of Sender Back-End Client Certificate

In this use case, the client certificate (of the sender) has to be renewed.

The following figure illustrates the communication path that is relevant for this use case.

---

**Note**

It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

Certificate renewal has to be performed in the following sequence.
1. Sender administrator: Creates new key pair/client certificate (also the root certificate (CA) may be changed).

2. Sender administrator: Provides tenant administrator with the certificate and the root certificate. Sender administrator has to make sure that the client certificate in the sender keystore is signed by one CA that is listed in Load Balancer Root Certificates Supported by SAP [page 214].

3. After the CA certificate was added to the truststore of the virtual server, tenant administrator forwards the certificate to the integration developer and asks him to add the subject and issuer DN of the certificate to the authorization interceptor of the inbound channel corresponding to the sender system.

4. Integration developer and tenant administrator need to perform the following steps:
   1. Integration developer: In all HTTP-based sender channels (like HTTP, SOAP, SAP XI, AS2) with Authorization setting Client Certificate adapts entries for the Subject DN and Issuer DN with the entries for the new client certificate, and redeploys the corresponding integration flows.
   2. Tenant administrator: If the involved integration flows use HTTP-based sender channels with Authorization setting User Role, the tenant administrator creates a new Certificate-to-User Mapping with the new certificate (making sure that the same user like in the entry with the old certificate is used).

5. Integration developer: Informs the tenant administrator that authorization interceptor has been configured with the new certificate.

6. Tenant administrator: Informs the sender administrator that he now can sent messages with the new client certificate.

7. Sender administrator: Configures sender system that way that it sends HTTPS messages with the new client certificate.

8. Sender administrator: Informs tenant administrator that sender system is using now the new client certificate for the HTTPS communication.

9. The tenant administrator needs to perform the following steps:
   1. If the involved integration flows use HTTP-based sender channels with Authorization setting Client Certificate (see step 4 a)), the tenant administrator informs the integration developer that he or she can remove the DNs of the old certificate from the authorization interceptor in the integration flows.
   2. If there is an entry in the Certificate-to-User Mappings with the old certificate, the tenant administrator removes this entry.

10. Integration developer: Removes the DNs of the old certificate from the authorization interceptor in the integration flow and redeploys the integration flow.

**i Note**

Steps 4-6 and 10-11 are only necessary if the subject DN or issuer DN of the certificate has been changed.

**Related Information**

Involved Roles [page 135]
5.1.3.1.3 Basic Authentication (Outbound)

Basic authentication uses credentials to allow the identification of trusted communication partners. Credentials are composed of user and password and, in case the SuccessFactors connector is involved, additional attributes. In addition to credentials, basic authentication also uses a one-way SSL connection which requires server certificates. Therefore, security artifact update involves both the update of credentials and of the involved certificates.

5.1.3.1.3.1 Renewal of User and Password

In this use case, the user (through which the tenant calls the receiver system) is replaced by a new user (and password) in the receiver system.

Note

It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

Security artifact renewal has to be performed in the following sequence:

1. Receiver administrator: Creates a new user (user1) and assigns authorization roles to the user. After this step has been performed, two users are configured on the receiver system for a certain HTTPS communication: the old user (user0) and the new one (user1).
2. Receiver administrator: Informs the tenant administrator that he wants to exchange the old user (user0) with a new user (user1). The new user also should have a new password.
3. Tenant administrator: Starts the Integration Operations user interface, opens the related User Credentials artifact (specified for user0 and the communication path with the receiver system) and exchanges the old user/password with the new user/password.
4. Tenant administrator: Restarts the corresponding integration flow(s) using the Integration Operations user interface.
5. Tenant administrator: Informs the receiver administrator that user/password has been exchanged.
6. Receiver administrator: Removes the old user.

In case a SuccessFactors receiver channel is used, note the following, slightly adapted sequence of steps:

1. Success Factors administrator (receiver administrator): Creates a new user/password and assigns the adequate authorizations to the user for the new company ID.
2. Success Factors administrator: Informs the tenant administrator that a new user/password for new company ID has been created and that the old user/password/company ID will be no longer valid from a certain point in time.
3. Tenant administrator: Changes the user/password/company ID in the Integration Operations user interface for the existing credentials (User Credentials artifact).
4. Success Factors administrator: Removes old user/password/company ID from Success Factors system after the specified point in time has been reached.
5.1.3.1.3.2 Renewal of Password Only

In this use case, the password of the user through which the tenant calls the receiver system is replaced in the receiver system.

i Note

It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

To exchange the password of a user without any downtime, the receiver administrator has to create an intermediate user as described for the use case Renewal of User and Password (without deleting the old user).

1. Receiver administrator: Creates a new intermediate user (user1) and assigns authorization roles to the user. After this step has been performed, two users are configured on the receiver system for a certain HTTPS communication: the old user (user0) and the new one (user1).
2. Receiver administrator: Informs the tenant administrator that he wants to change the password of a certain user used for a certain HTTPS communication and that he has created an intermediate user (user1) with a certain password.
3. Tenant administrator: Starts the Integration Operations user interface, opens the related User Credentials artifact (specified for user0 and the communication path with the receiver system) and exchanges the old user0/password with the new user1/password.
4. Tenant administrator: Informs the receiver administrator that the client now uses the intermediate user (user1).
5. Receiver administrator: Changes the password of the original user (user0).
6. Receiver administrator: Informs the tenant administrator that the password of the original user (user0) has been changed.
7. Tenant administrator: Starts the Integration Operations user interface, opens the related User Credentials artifact (now containing the credentials of user1) and exchanges the credentials (user/password) of the intermediate user (user1) with the credentials (user name and new password) of the original user (user0).
8. Tenant administrator: Informs the receiver administrator that user password has been changed.
9. Receiver administrator: Removes the intermediate user.

i Note

In case the receiver administrator does not accept this complicated procedure, a simplified procedure might be adopted that way that the tenant administrator just changes the password as soon as he notices that the connection to the receiver system fails due to a wrong password.
The same procedure is applicable in case a SuccessFactors receiver channel is used.

Related Information
Involved Roles [page 135]

5.1.3.1.4 Basic Authentication (Inbound)

5.1.3.1.4.1 Renewal of User and Password

In this use case, the user (through which a sender calls the tenant) is replaced by a new user (and password) on the SAP cloud platform.

Security artifact renewal has to be performed in the following sequence:
1. SAP: Informs the sender administrator that the sender back-end system should use new user/password for communication with the tenant.
2. Sender administrator: Changes the user and password in the HTTPS sender client (sender back-end).
3. Sender administrator: Informs SAP that user has been changed in the sender client.

5.1.3.1.4.2 Renewal of Password Only

In this use case, the password of the user (through which the sender system is supposed to call the tenant) is replaced by a new password on the SAP cloud platform.

To exchange the password of a user without any downtime, SAP has to create an intermediate user as described for the use case Renewal of User and Password (without deleting the old user).

Security artifact renewal has to be performed in the following sequence:
1. SAP: Informs the sender administrator that he wants to change the password of a certain user used for HTTPS communication with the tenant and that he has created an intermediate user (user1) and password.
2. Sender administrator: Exchanges the old user/password (user0) with the intermediate user/password (user1) in the HTTPS sender client (back-end system).
3. Sender administrator: Informs SAP that the sender client now uses the intermediate user (user1).
4. SAP: Informs the sender administrator that the password of the original user (user0) has been changed.
5. Sender administrator: Exchanges the user/password of the intermediate user (*user1*) with the original user (*user0*) (and with the new password).

6. Sender administrator: Informs SAP that user and password has been changed.

### 5.1.3.2 Security Artifact Renewal for SFTP-Based Communication

Using SSH File Transfer Protocol (SFTP), the basic set up is that an SFTP client is connected to an SFTP server from which the client pulls data or to which the client pushes data. Secure SFTP communication is enabled by the usage of public/private key pairs as well as a trust relationship between client and server implemented by known_hosts files.

In scenarios using SFTP, the tenant is always an SFTP client either pushing files to the SFTP server or pulling files from it.

Where the SFTP is hosted, depends on the scenario.

In SFTP security artifact renewal processes, the following roles are involved:

- SFTP server administrator
- Tenant administrator (is always the SFTP client administrator)
- Integration developer

The following security artifacts can be subject to change and underly a renewal process:

- Public/private key pair (of either SFTP server or tenant)
- User who either pushes files to SFTP server or pulls files from it

### 5.1.3.2.1 Renewal of the SFTP Server Key

In this use case, an SSH key pair is renewed on the SFTP server.

**Note**

It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

Security artifact renewal has to be performed in the following sequence:

1. SFTP server administrator: Creates new server key pair.
2. SFTP server administrator: Provides tenant administrator with the public key and informs him that the key will be exchanged on the SFTP server at a certain point in time.
3. Tenant administrator: Adds the new public key to the known_hosts file.
   After that step has been performed, the client (tenant) trusts the server either he has the old or new key.
4. Tenant administrator: Removes the old public key entry form the known_hosts file after the agreed point in time.
5.1.3.2.2 Renewal of the SFTP Client Key (on Tenant)

In this use case, an SSH key pair is renewed on the SFTP client (tenant).

**Note**

It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Security artifact renewal has to be performed in the following sequence:

1. Tenant administrator: Generates new key pair for the SFTP client.
2. Tenant administrator: Exports the public key from the keystore and provides the SFTP server administrator with the public key.
3. SFTP server administrator: Imports the public key (provided by the tenant administrator) into the SFTP server truststore.
4. SFTP server administrator: Informs tenant administrator that public key has been imported into the truststore of the SFTP server.
5. Tenant administrator: Exchanges in the keystore the old key pair with the new one.

5.1.3.2.3 Renewal of User on SFTP Server

Files are stored on the SFTP server in directories referred to as mailboxes. For each mailbox, a user is specified to control access to the data. In this use case, the mailbox user on the SFTP server is changed.

**Note**

It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

There are two sub use cases, depending on whether the tenant pulls data from or pushes data to the SFTP server.
Tenant Pulls Data from SFTP Server

User renewal has to be performed in the following sequence:

1. SFTP server administrator: Creates a new user on the SFTP server with all relevant configurations (for example, mailbox).
2. SFTP server administrator: Informs the tenant administrator that an old user shall be exchanged by a new one and that the new user already exists on the SFTP server.
3. Tenant administrator: Informs integration developer that he should exchange the old SFTP user with the new one (in the corresponding integration flow).
4. Integration developer: Exchanges the old SFTP user with the new one in the integration flow.
5. Integration developer: Informs the tenant administrator that user has been exchanged.
6. Tenant administrator: Informs SFTP server administrator that the user has been exchanged.
7. SFTP server administrator: Makes sure that all data of the old user has been fetched by the tenant. If this is not the case he transfers the relevant data into the mailbox of the new user.

Tenant Pushes Data to SFTP Server

User renewal has to be performed in the following sequence:

1. SFTP server administrator: Creates a new user on the SFTP server with all relevant configurations (for example, mailbox).
2. SFTP server administrator: Informs the tenant administrator that an old user shall be exchanged by a new one and that the new user already exists on the SFTP server.
3. Tenant administrator: Informs the integration developer that he should exchange the old SFTP user with the new one (in the corresponding integration flow).
4. Integration developer: Exchanges the old SFTP user with the new one in the integration flow.
5. Integration developer: Informs the tenant administrator that user has been exchanged.
6. Tenant administrator: Informs the SFTP server administrator that the user has been exchanged.
7. If a pulling component relies on the data, the SFTP server administrator makes sure that the poller has read all data from the mailbox of the old user. If this is not the case, the SFTP server administrator transfers the data into the mailbox of the new user.

Related Information

Involved Roles [page 135]

5.1.4 Security Artifact Renewal for Message Encryption/Signature
5.1.4.1 Security Artifact Renewal for PKCS#7/CMS

5.1.4.1.1 Renewal of Keys for CMS/PKCS#7 Signer - Outbound

This use case covers all situations where private keys (used by the tenant to sign outbound messages) are changed. The renewal process ensures that the related public verification key is changed at the receiver side that way that no downtime is required.

The signer (when configured to use the CMS/PKCS#7 standard) uses one or more private keys to sign a single payload. These private keys are provided in the outbound keystore of the tenant. The resulting signed data object can contain several signatures from different private keys. To locate the different private keys in the keystore, aliases can be specified in the corresponding integration flow signer step.

The following figure illustrates the communication path that is relevant for this use case.

Legend
- Tenant private key
- Tenant public key

i Note
It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and
integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

The renewal process depends on whether the receiver system can verify signed data with different public keys at one point in time.

**Receiver is able to Verify Payloads Signed by the old Key and Payloads Signed by the new Key at the Same Time**

This renewal process implies the following sequence of steps.

1. Tenant administrator: Creates a new key pair.
2. Tenant administrator: Provides the new certificate to the receiver administrator.
3. Receiver administrator: Configures the receiver system that way that it is able to verify payloads signed by the old key or payloads signed by the new key.
4. Receiver administrator: Informs the tenant administrator that the receiver system is able to verify payloads signed by the old key or payloads signed by the new key.
5. Tenant administrator: Exchanges the old key pair with the new key pair, keeping the old alias. From now on outbound messages are signed with the new key.
6. Tenant administrator: Informs the receiver administrator that the key pair has been exchanged.
7. Receiver administrator: Removes the old key pair. From now on, the receiver system can only verify payloads signed by the new key.

**Receiver is only able to Verify Payloads Signed by the Same Key at one Point in Time**

This renewal process requires cooperation of tenant administrator, integration developer and receiver administrator.

**i Note**

It is assumed that the receiver system can manage CMS/PKCS#7-signed data containing several signatures. This should be the case because the specification requires it.

1. Tenant administrator: Creates a new key pair.
2. Tenant administrator: Adds the new key pair to the keystore with a new alias.
3. Tenant administrator: Informs the integration developer.
4. Integration developer: Changes the integration flow. In particular, the integration developer adds the new alias to the Signer Parameters table of the CMS/PKCS7 Signer step. All other parameters like Signature Algorithm, Include Certificates, Include Signing Time have to be the same as specified for the entry with the old alias. From now on, the tenant sends signed data containing two signatures both from the old and the new key.
5. Tenant administrator: Provides the new certificate to the receiver administrator and informs him that the tenant is sending from now on signed data containing two signatures, a signature of the old key and a signature of the new key.
6. Receiver administrator: Exchanges the key pair that way that the receiver system verifies from now on the signature of the new key.
7. Receiver administrator: Informs the tenant administrator that the receiver system verifies the signature of the new key.
8. Tenant administrator: Informs the integration developer that he can remove the old alias from the integration flow signer step.
9. Integration developer: Removes the old alias from the integration flow signer step.
10. Integration developer: Informs the tenant administrator that the alias has been removed.
11. Tenant administrator: Removes the old key pair from the keystore.

**Receiver can only Verify Payloads Signed by the Same Key at one Point in Time but Accepts PKCS7/CMS Data with two Signatures**

This use case is supported as of release 1.7 of the Integration Designer.

This procedure is not applicable for systems based on AS ABAP.

The following assumptions apply:
- The receiver system can handle CMS/PKCS7-signed data containing several signatures. Actually this should be the case because this is part of the specification. Note that systems based on AS ABAP do not support this.
- The PKCS7/CMS signer step contains two aliases, one for the current private key and one for the new private key.
  If this is not the case, the integration developer has to be asked to add a second alias.

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Adds the new certificate to the keystore, taking into account that the alias is used which is specified in the PKCS7/CMS signer step for the new certificate.
   From now on the PKCS7/CMS data format contains two signatures.
3. Tenant administrator: Provides the receiver administrator with the new certificate.
4. Receiver administrator: Exchanges the old certificate with the new certificate and performs relevant configuration steps.
5. Receiver administrator: Informs the tenant administrator that certificate has been exchanged.
6. Tenant administrator: Removes the old certificate from the keystore.

**Receiver can only Verify Payloads Signed by the Same Key at one Point in Time and Accepts only PKCS7/CMS Data with Exactly one Signature**

This procedure is applicable for systems based on AS ABAP.

> **Note**
> This procedure implies a downtime.
1. Tenant administrator: Creates new key pair/certificate.
2. Tenant administrator and receiver administrator: Agree on a downtime.
3. Tenant administrator: Provides receiver admin with the certificate.
4. During the downtime:
   1. Tenant administrator: Exchanges key pair/certificate in keystore, keeping the old alias.
   2. Receiver administrator: Exchanges the certificate in receiver system.

Related Information

How PKCS#7 Works [page 223]
Involved Roles [page 135]

5.1.4.1.2 Renewal of Keys for CMS/PKCS#7 Verifier - Inbound

This use case covers all situations where private keys used by a sender to sign messages sent to the tenant (in our terminology: inbound messages) are changed. The renewal process ensures that the related public verification key is changed at the tenant side that way that no downtime is required.

The verifier (specified in the integration flow to use the CMS/PKCS#7 standard) uses a public key to verify a payload signed by the sender. This public key has been imported into the tenant keystore as X509 certificate during the onboarding process. The verifier uses an alias configured in the corresponding integration flow step to locate the public key in the keystore. The renewal process depends on whether the sender system can send signed data with signatures from several keys. The CMS/PKCS#7 specification does allow this.

The following figure illustrates the communication path that is relevant for this use case.

Legend

- Sender private key
- Sender public key

Note

It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and
Sender is able to Send Payload Signed by Old and New Key

1. Sender administrator: Creates a new key pair.
2. Sender administrator: Configures the sender system that from now on it sends signed data with two signatures, from the old and new key.
3. Sender administrator: Provides new public key (certificate) to the tenant administrator.
4. Tenant administrator: Exchanges the old public key with the new one in the keystore, keeping the old alias. (From now on the avatar verifies the signature of the new key.)
5. Tenant administrator: Informs the sender administrator that the key has been exchanged.
6. Sender administrator: Configures the sender system that way that a payload with one signature from the new key is being sent.
7. Sender administrator: Removes the old key pair.

Sender is Only able to Send Payload Signed by One Key

This procedure applies for Integration Designer as of Version 1.7.

For the renewal process it is assumed that the verifier integration flow step contains two aliases, one for the current certificate and one for the new certificate.

If the alias for the new certificate does not yet exist, the integration developer must add such an alias.

1. Sender administrator: Creates a new key pair.
2. Sender administrator: Provides the new public key to the tenant administrator.
3. Tenant administrator: Adds the new public key (certificate) to the keystore, taking into account that the alias is used which is specified in the verifier step for the new certificate.
4. Tenant administrator: Informs the sender administrator that payloads signed with the new key can be sent.
5. Sender administrator: Configures the sender system that way that from now on it sends payloads signed with the new key.
6. Sender administrator: Removes the old key pair.
7. Sender administrator: Informs the tenant administrator about the fact that the old key pair has been removed.
8. Tenant administrator: Removes the old public key (certificate) from the keystore after a time period which corresponds to the guaranteed delivery time (to make sure that payloads signed with the old key are no longer in the SAP cloud system).

Related Information

How PKCS#7 Works [page 223]
5.1.4.1.3 Renewal of Keys for CMS/PKCS#7 Encryptor - Outbound

This use case covers all situations where a private key used for message decryption is changed by a receiver. The renewal process ensures that the related public encryption key is changed at the tenant side that way that no downtime is required.

The encryptor (specified in the integration flow to use the CMS/PKCS#7 standard) uses one or serveral public keys to encrypt a payload. The resulting enveloped data then contains one or several recipient information elements corresponding to the public keys. These recipient information elements contain information about the certificates corresponding to the public keys (issuer DN and serial number of the certificate). The encryptor uses aliases configured in the integration flow step to loacte the certificates in the keystore.

The following figure illustrates the communication path that is relevant for this use case.

**Legend**
- 🔑 Receiver private key
- 🔑 Receiver public key

**i Note**
It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

Receiver is able to Decrypt Payloads Encrypted by the Old Key and Payloads Encrypted by the New Key at one Point in Time

1. Receiver administrator: Creates a new key pair.
2. Receiver administrator: Configures receiver system that way that it either can verify payloads encrypted with the old key or payloads encrypted with the new key.
3. Receiver administrator: Provides the new public to the tenant administrator.
4. Tenant administrator: Exchanges the public key (certificate) with the new public key, keeping the old alias. From now on, messages are being encrypted with the new key.
5. Tenant administrator: Informs the receiver administrator that certificate has been exchanged.
6. Receiver administrator: Removes the old key pair.

**Receiver is only able to Decrypt Payloads Encrypted by the Same Key at one Point in Time**

*Note*

It is assumed that receiver system can manage CMS/PKCS7 enveloped data containing several encryption recipients.

1. Receiver administrator: Creates a new key pair.
2. Receiver administrator: Provides the public key to the tenant administrator.
3. Tenant administrator: Adds the new public key to the keystore with a new alias.
4. Tenant administrator: Informs the integration developer that new certificate with new alias has been added for encryption usage.
5. Integration developer: Changes the integration flow.
   The integration developer adds the new alias to the Receiver Public Key Alias list of the CMS/PKCS7 Encryptor step.
   After this step has been performed, the tenant sends CMS/PKCS7 enveloped data containing two recipient information elements: from the old and from the new certificate.
6. Tenant administrator: Informs the receiver administrator that the tenant sends CMS/PKCS7 enveloped data with two recipient information elements: from the old and new certificate.
7. Receiver administrator: Exchanges the certificate that way the receiver system now uses the recipient information of the new certificate to decrypt the payload.
8. Receiver administrator: Removes the old key pair.
9. Receiver administrator: Informs the tenant administrator that the receiver now uses the recipient information of the new certificate to decrypt the payload.
10. Tenant administrator: Informs the integration developer that the old alias can be removed from the integration flow encryptor step.
11. Integration developer: Removes the old alias from the integration flow encryptor step.
    After this step has been performed, the sent CMS/PKCS7 enveloped data does only contain one recipient information of the new certificate.
12. Integration developer: Informs the tenant administrator that the alias has been removed.
13. Tenant administrator: Removes the old certificate from keystore.
Receiver can only Decrypt Payloads Encrypted by the Same Key at one Point in Time and Accepts PKCS7/CMS-Enveloped Data Containing Symmetric Keys Encrypted by Several Asymmetric Keys

This procedure is supported as of release 1.7 of the Integration Designer.

This procedure is applicable for systems based on AS ABAB.

The following assumptions apply:

- The receiver system can handle with CMS/PKCS7-enveloped data containing several encryption recipients. This should be the case because this is part of the specification.
- The PKCS7/CMS encryptor step contains two aliases, one for the current public key and one for the new public key. If this is not the case, the integration developer has to be asked to add a second alias.

1. Receiver administrator: Creates new key pair/certificate.
2. Receiver administrator: Provides new certificate to the tenant administrator.
3. Tenant administrator: Adds new certificate to the keystore taking into account that the alias is used which is specified in the PKCS7/CMS encryptor step for the new certificate. From now on the PKCS7/CMS enveloped data contains two encryptions of the symmetric key.
4. Tenant administrator: Informs the receiver administrator that PKCS7/CMS-enveloped data are sent with two encryptions for the symmetric key.
5. Receiver administrator: Exchanges old key pair/certificate with the new one.
6. Receiver administrator: Informs the tenant administrator that the certificate has been exchanged.
7. Tenant administrator: Removes old certificate from the keystore.

Receiver can Only Decrypt Payloads Encrypted by the Same Key at one Point in Time and Does not Accept PKCS7/CMS-Enveloped Data Containing Symmetric Keys Encrypted by Several Asymmetric Keys

This procedure implies a downtime.

1. Receiver administrator: Creates new key pair/certificate.
2. Receiver administrator and tenant administrator: Agree on a downtime.
4. During the downtime:
   1. Receiver administrator: Exchanges the old key pair/certificate with the new one.
   2. Tenant administrator: Exchanges the old certificate with the new one, keeping the old alias

Related Information

How PKCS#7 Works [page 223]
Involved Roles [page 135]
5.1.4.1.4 Renewal of Keys for CMS/PKCS#7 Decryptor - Inbound

This use case covers all situations where private keys used by the tenant to decrypt messages from a sender (in our terminology: inbound messages) are changed. The renewal process ensures that the related public encryption key is changed at sender side that way that no downtime is required.

The CMS/PKCS7 decryptor uses a private key to decrypt a PKCS7/CMS encrypted payload. This private key is provided in the tenant keystore together with a X509 certificate. The decryptor uses an alias configured in the corresponding integration flow step to locate the private key in the keystore.

The following figure illustrates the communication path that is relevant for this use case.

![Communication Path Diagram]

Legend
- Tenant private key
- Tenant public key

i Note
It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

Sender is able to Encrypt Payload with the old Key and the new Key

1. Tenant administrator: Creates a new key pair.
2. Tenant administrator: Adds the new key pair and the corresponding certificate into the keystore. After this step has been performed, the decryptor accepts payloads encrypted by the old and the new public key.
3. Tenant administrator: Hands over the new certificate to the sender administrator.
4. Sender administrator: Exchanges the certificate.
5. Sender administrator: Informs the tenant administrator.
6. Tenant administrator: Removes the old key pair from the keystore after a time period (which at least corresponds to the guaranteed delivery time) so that he can be sure that no payload encrypted with the old public key has been sent.
**Sender is only able to Encrypt Payload with one Key**

This process is applicable as of version 1.6 of the Integration Designer.

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Adds the new key pair and the corresponding certificate into the keystore. From now on, the decryptor accepts payloads encrypted by the old and new public key.
3. Tenant administrator: Hands over the new certificate to the sender administrator.
4. Sender administrator: Exchanges the certificate.
5. Sender administrator: Informs the tenant administrator about the preceding step.
6. Tenant administrator: Removes the old key pair/certificate from the payload security keystore after a time period (which at least corresponds to the guaranteed delivery time) so that he can be sure that no payload encrypted with the old public key is being sent.

**Related Information**

- How PKCS#7 Works [page 223]
- Involved Roles [page 135]

**5.1.4.2 Security Artifact Renewal for OpenPGP**

**Related Information**

- How OpenPGP Works [page 227]
- Renewal of OpenPGP Encryption Key - Outbound [page 161]
- Renewal of OpenPGP Encryption Key - Inbound [page 164]
- Renewal of OpenPGP Signer Key - Outbound [page 165]
- Renewal of OpenPGP Signer Key - Inbound [page 167]
5.1.4.2.1 Renewal of OpenPGP Encryption Key - Outbound

In this use case, the tenant is the sender (outbound communication) and the receiver renews the encryption key.

The following figure illustrates the communication path that is relevant for this use case:

![Diagram](image)

Legend
- Receiver private key
- Receiver public key

**i Note**

It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

The renewal process depends on the capabilities of the customer receiver system.

**Receiver Can Decrypt Payloads Encrypted by the Old Key and Payloads Encrypted by the New Key at One Point in Time**

This use case requires a change of the integration flow if the user ID changes.

1. Receiver administrator: Creates a new PGP key pair.
2. Receiver administrator: Configures receiver system so that it can decrypt either payloads encrypted with the old key or payloads encrypted with the new key.
3. Receiver administrator: Provides the tenant administrator with the new PGP public key through a secure channel and informs the tenant administrator that for a certain period of time the receiver system will accept PGP messages encrypted either with the old key or with the new key.
4. Tenant administrator: Imports the new PGP public key into the PGP Public Keyring. If the public key has not been received through a secure channel, the tenant administrator checks that the new imported key has the correct fingerprint by comparing the fingerprint with a fingerprint provided via a trustworthy channel (phone, signed e-mail).
5. Tenant administrator: Informs the integration developer that he has to exchange the old encryption user ID with the new encryption user ID in the PGP encryptor step (only if the new key has a different user ID to the old key).
6. Integration developer adapts the integration flow (only if the new key has a different user ID to the old key). The integration developer adds the new user ID (or a part of the user ID) to the list of user IDs in the PGP encryptor step. The adapted integration flow has to be newly deployed. The old user ID must still be kept on the user ID list.

7. Tenant administrator: Deletes the old public key from the PGP Public Keyring and deploys the changed PGP Public Keyring. From now on the payloads are encrypted with the new public key.

8. After the agreed time period, the receiver administrator removes the old key pair and configures the receiver system so that from now on it can only receive payloads encrypted by the new key.

Receiver Can Only Decrypt Payloads Encrypted with the Same Key at One Point in Time and Accepts PGP Messages Containing Symmetric Keys Encrypted with Several Asymmetric Keys

This use case requires a change of the integration flow if the user ID changes.

1. Receiver administrator: Creates new PGP key pair.

2. Receiver administrator: Provides tenant administrator with the new PGP public key and informs the tenant administrator that the receiver system will only be able to decrypt PGP messages that are encrypted with the new key as of a certain date. Note that the PGP message may still contain an additional package containing the symmetric key encrypted with the old key.

3. Tenant administrator: Imports the new PGP public key into the PGP Public Keyring and checks that the new imported key has the correct fingerprint by comparing the fingerprint with a fingerprint provided via a trustworthy channel (phone, signed e-mail). This step is only necessary if the public key has not been received through a secure channel.

4. Tenant administrator: Informs the integration developer that he has to exchange the old encryption user ID with the new encryption user ID in the PGP encryptor step (only if the new key has a different user ID to the old key).

5. Integration developer adapts the integration flow (only if the new key has a different user ID to the old key). The integration developer adds the new user ID (or a part of the user ID) to the list of user IDs in the PGP encryptor step. The adapted integration flow has to be newly deployed. The old user ID must still be kept on the user ID list.

6. Tenant administrator: Deploys the changed PGP Public Keyring. From now on the payload is encrypted with the old key and the new key.

7. At the specified date the receiver administrator removes the old key from the receiver system.

8. After the specified date the tenant administrator removes the old public key from the PGP Public Keyring and deploys the changed PGP Public Keyring. From now on the payload is only encrypted with the new key.
Receiver Can Only Decrypt Payloads Encrypted with the Same Key at One Point in Time and Does Not Accept PGP Messages Containing Symmetric Keys Encrypted with Several Asymmetric Keys

This use case requires a downtime.
1. Receiver administrator: Creates new PGP key pair.
2. Receiver administrator and tenant administrator agree on a downtime.
3. Receiver administrator: Provides tenant administrator with the new PGP public key.
4. Tenant administrator: Imports the new public key into the PGP Public Keyring and checks that the fingerprint of the new PGP public key is correct by comparing the fingerprint with a fingerprint provided via a trustworthy channel (phone, signed e-mail). This step is only necessary if the public key was not received through a secure channel.
5. During downtime, the following happens:
   1. Tenant administrator: Informs the integration developer that he has to exchange the old encryption user ID with the new encryption user ID in the PGP encryptor step (only if the new key has a different user ID to the old key).
   2. Integration developer adapts the integration flow (only if the new key has a different user ID to the old key).
      The integration developer adds the new user ID (or a part of the user ID) to the list of user IDs in the PGP encryptor step.
      The adapted integration flow has to be newly deployed.
   3. Tenant administrator: Removes the old key from the PGP Public Keyring and deploys the PGP Public Keyring.
   4. Receiver administrator: Exchanges the old key with the new key in the receiver system.

Related Information

Involved Roles [page 135]
5.1.4.2.2 Renewal of OpenPGP Encryption Key - Inbound

In this use case, the tenant is the receiver (inbound communication) and renews the encryption key.

The following figure illustrates the communication path that is relevant for this use case:

![Communication Path Diagram]

**Legend**
- ☑ Tenant private key
- ☑ Sender public key

**i Note**
It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

1. Tenant administrator: Creates a new PGP key pair with the same user ID as the old key in the PGP Secret Keyring.
2. Tenant administrator: Deploys the changed PGP Secret Keyring.
   From this moment on the PGP decryptor accepts payloads encrypted with the old key or payloads encrypted with the new public key.
3. Tenant administrator: Exports the new PGP public key.
4. Tenant administrator: Provides the sender administrator with the exported key via a secure channel and informs the sender administrator about the following:
   - For a certain period of time the tenant will be able to accept payloads encrypted with the old key or payloads encrypted with the new public key.
   - After this period only payloads encrypted with the new key are accepted.
5. Sender administrator: Exchanges the old key with the new key, so that from now on the sender system sends payloads encrypted with the new key.
6. After the specified period is over, the tenant administrator removes the old key pair from the PGP Secret Keyring and deploys the PGP Secret keyring.

**Related Information**

*Involved Roles [page 135]*
5.1.4.2.3 **Renewal of OpenPGP Signer Key - Outbound**

In this use case, the tenant is the sender (outbound communication) and renews the signer key.

The following figure illustrates the communication path that is relevant for this use case:

![Communication Path Diagram]

**Legend**

- Tenant private key
- Receiver public key

**i Note**

It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

The renewal process depends on the capabilities of the customer receiver system.

### Receiver Can Verify Payloads Signed with the Old Key and Payloads Signed with the New Key at One Point in Time

1. **Tenant administrator**: Creates a new PGP key pair in the tenant’s PGP Secret Keyring with the same user ID as the old key (without deploying the changed PGP Secret Keyring yet).
2. **Tenant administrator**: Exports the new PGP public key and provides receiver administrator with the new PGP public key via a secure channel. The tenant administrator informs the receiver administrator that as of a certain date the tenant will send payloads signed with the new key.
3. **Receiver administrator**: Configures the receiver system so that it can verify payloads signed with the old key or payloads signed with the new key.
4. On the specified date, the tenant administrator removes the old key from the PGP Secret Keyring and deploys the changed PGP Secret Keyring on the tenant. From now on the payloads are signed with the new key.
5. After the specified date, the receiver administrator removes the old key so that from now on the receiver system can only verify payloads signed by the new key.
Receiver Can Only Verify Payloads Signed with the Same Key at One Point in Time but Accepts PGP Messages with Two Signatures

1. Tenant administrator: Creates a new PGP key pair in the tenant’s PGP Secret Keyring with the same user ID as the old key.
2. Tenant administrator: Deploys the changed PGP Secret Keyring on the tenant. From now on, the signed PGP message will contain two signatures, one from the old key and one from the new key.
3. Tenant administrator: Exports the new PGP public key and provides the receiver administrator with the new public key and informs the receiver administrator about the following:
   ○ For a certain period of time, PGP messages with two signatures will be sent.
   ○ After this period, PGP messages with one signature made by the new key will be sent
4. Before the specified period ends, the receiver administrator exchanges the old key with the new key and configures the receiver system so that it now verifies the signature with the new key.
5. After the specified period the tenant administrator removes the old key from the PGP Secret Keyring and deploys the changed PGP Secret Keyring on the tenant. From now on, PGP messages signed by the new key are sent.

Receiver Can Only Verify Payloads Signed with the Same Key at One Point in Time and Accepts Only PGP Messages with Exactly One Signature

This use case requires a downtime.

1. Tenant administrator: Creates a new PGP key pair in the tenant’s PGP Secret Keyring with the same user ID as the old key.
2. Tenant administrator and receiver administrator agree on a downtime.
3. Tenant administrator: Exports the new public key from the PGP Secret Keyring and provides the receiver administrator with the exported public key.
4. During downtime, the following happens:
   1. Tenant administrator: Removes the old key from the PGP Secret Keyring and deploys the changed PGP Secret Keyring on the tenant.
   2. Receiver administrator: Exchanges the old key with the new key in the receiver system.

Related Information

Involved Roles [page 135]
5.1.4.2.4 Renewal of OpenPGP Signer Key - Inbound

In this use case, the tenant is the receiver (inbound communication) and the sender renews the signer key.

The following figure illustrates the communication path that is relevant for this use case:

![Communication Path Diagram]

**Legend**
- Sender private key
- Sender public key

**i Note**
It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

1. **Sender administrator**: Creates a new PGP key pair for signing.
2. **Sender administrator**: Provides the tenant administrator with the new public key and informs the tenant administrator that as of a certain the payloads are signed with the new key.
3. **Tenant administrator**: Imports the public key into the PGP Public Keyring of the tenant and checks that the fingerprint of the new PGP public key is correct by comparing the fingerprint with a fingerprint provided via a trustworthy channel (phone, signed e-mail).
   This step is only necessary if the public key has not been received through a secure channel.
4. **Tenant administrator**: Informs the integration developer that he has to exchange the old encryption user ID with the new encryption user ID in the PGP decryptor step (only if the new key has a different user ID to the old key).
5. **Integration developer** adapts the integration flow (only if the new key has a different user ID to the old key).
   The integration developer adds the new user ID (or a part of the user ID) to the list of user IDs in the PGP decryptor step (so that the old and new user ID are contained in the list of signer user IDs).
   The adapted integration flow has to be newly deployed. The old user ID must still be kept on the user ID list.
6. **Tenant administrator**: Deploys changed PGP Public Keyring on the tenant.
   From now on, the tenant accepts payloads signed with the old key or with the new key.
7. On the specified date, the sender administrator configures the sender system so that it sends payloads signed with the new key from now on. The sender administrator removes the old key pair.
8. After the specified date, the tenant administrator removes the old public key from the PGP Public Keyring and deploys the changed PGP Public Keyring on the tenant.
   From now on the tenant only accepts payloads signed with the new key.
Related Information

Involved Roles [page 135]

5.1.4.3 Security Artifact Renewal for XML Digital Signature

5.1.4.3.1 Renewal of Keys for XML Digital Signature Signer - Outbound

This use case covers all situations where private keys (used by the tenant to sign outbound messages based on XML Digital Signature) are changed. The renewal process ensures that the related public verification key is changed at the receiver side that way that no downtime is required.

i Note

It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

The signer (when configured to use the XML Digital Signature standard) uses a private key to sign a payload. The private key is provided in the outbound keystore of the tenant. To locate the private key in the keystore, an alias is specified in the corresponding integration flow signer step.

The renewal process depends on whether the receiver system can verify signed data with different public keys at one point in time.

Receiver is able to Verify Payloads Signed by the old Key and Payloads Signed by the new Key at the Same Time

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Provides the receiver administrator with the new certificate.
3. Receiver administrator: Configures the receiver system so that the receiver system can verify payloads signed by the old key or payloads signed by the new key.
4. Receiver administrator: Informs the tenant administrator that the receiver system can verify payloads signed by the old key or payloads signed by the new key.
5. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate in the keystore, keeping the old alias.
   From now on, the message is signed with the new key.
6. Tenant administrator: Informs the receiver administrator that certificate has been exchanged.
7. Receiver administrator: Removes the old certificate that way that from now on the receiver system can only verify payloads signed by the new key.

**i Note**

This is the same process as to be applied for the CMS/PKCS#7 Signer.

---

**Receiver is only able to Verify Payloads Signed by the Same Key at one Point in Time**

This process implies a downtime.

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Provides the receiver administrator with the new certificate.
3. Tenant administrator: Agrees with the receiver administrator on a downtime. During certificate exchange no signed message is sent.
4. Tenant administrator: Informs the integration developer that the integration flow which signs the payload shall be undeployed.
5. Integration developer: Undeploys the integration flow (using the *Deployed Artifacts* editor tab of the Integration Operations feature). From now on, no further signed message is sent to the receiver system.
6. Integration developer informs tenant administrator about the preceding step.
7. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate, keeping the old alias in the keystore. From now on, the data are signed with the new key.
8. Tenant administrator: Informs the receiver administrator that no signed messages are being sent.
9. Receiver administrator: Exchanges the certificate in the receiver system.
10. Receiver administrator: Informs the tenant administrator that from now on the receiver system expects payloads signed by the new key.
11. Tenant administrator: Informs the integration developer that he can redeploy the integration flow.
12. Integration developer: Redeploys the integration flow (using the *Deployed Artifacts* editor tab of the Integration Operations feature).

---

**Related Information**

[How XML Signature Works](#) [page 225]

[Involved Roles](#) [page 135]

### 5.1.4.3.2 Renewal of Keys for XML Digital Signature Verifier - Inbound

This use case covers all situations where private keys used by a sender to sign messages sent to the tenant (in our terminology: inbound messages) based on XML Digital Signature are changed. The renewal process
ensures that the related public verification key is changed at the tenant side that way that no or minimum downtime is required.

<table>
<thead>
<tr>
<th>i Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>It depends on the <strong>operating model</strong> whether the tenant administrator and the integration developer are at the customer or at SAP. In the <strong>customer-managed operating model</strong>, the tenant administrator and integration developer tasks are performed by the customer. In the <strong>SAP-managed operating model</strong>, these tasks are performed by SAP.</td>
</tr>
</tbody>
</table>

The verifier (specified in the integration flow to use the XML Digital Signature standard) uses a public key to verify a payload signed by the sender. This public key has been imported into the tenant keystore as X509 certificate. The verifier uses an alias configured in the corresponding integration flow step to locate the public key in the keystore.

To apply the following process, the following prerequisites are met:

- The Integration Designer as of version 1.7 is used.
- The sender sends in the XML Signature information about the signer certificate (certificate, whole certificate chain, issuer DN and serial number of certificate, or combinations).
- The XML Signature Verifier step (in the integration flow) contains two aliases, one for the current certificate and one for the new certificate.

If the alias for the new certificate does not yet exist in the XML Signature step then the integration developer has to add such an alias prior to the renewal process.

1. Sender administrator: Creates new key air/certificate.
2. Sender administrator: Provides the tenant administrator with the new certificate.
3. Tenant administrator: Adds the new certificate to the keystore taking care that for the alias the alias is used which already has been specified in the XML Signature Verifier step (of the related integration flow).
4. Tenant administrator: Informs the sender administrator that payloads signed with the new key can be sent.
5. Sender administrator: Configures sender system that way that from now on payloads signed with the new private key are being sent.
6. Sender administrator: Removes the old key pair/certificate.
7. Sender administrator: Informs the tenant administrator that the sender systems sends payloads signed with the new key.
8. Tenant administrator: Removes the old certificate from the payload security keystore after a certain time period (which at least corresponds to the guaranteed delivery time).

The time period is necessary to make sure that payloads signed with the old key are no longer in the SAP cloud system.

**Related Information**

- How XML Signature Works [page 225]
- Involved Roles [page 135]
5.1.4.4 Security Artifact Renewal for WS-Security

Web services allow you to implement the request-response pattern. Web-Services Security (WS-Security) allows you to protect the request and response messages with digital signatures and encryption.

For Web services scenarios, there are separate security artifact renewal use cases for both request and response messages. WS-Security settings are configured in the sender and receiver SOAP adapters - depending on whether the tenant is a Web services provider or Web services consumer.

Inbound Communication (Tenant as WS Provider)

The following figure illustrates the communication paths that have to be considered when using WS-Security.

The corresponding settings on the tenant side are configured in the sender SOAP adapter.

The table below contains the security renewal use cases:

<table>
<thead>
<tr>
<th>Use Case</th>
<th>More information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tenant verifies inbound request message.</td>
<td>Security Artifact Renewal for WS-Security (Tenant Verifies Inbound Request) [page 173]</td>
</tr>
<tr>
<td>Tenant decrypts inbound request message.</td>
<td>Security Artifact Renewal for WS-Security (Tenant Decrypts Inbound Request) [page 174]</td>
</tr>
<tr>
<td>Tenant signs inbound response message.</td>
<td>Security Artifact Renewal for WS-Security (Tenant Signs Inbound Response) [page 175]</td>
</tr>
<tr>
<td>Tenant encrypts inbound response message.</td>
<td>Security Artifact Renewal for WS-Security (Tenant Encrypts Inbound Response) [page 177]</td>
</tr>
</tbody>
</table>
Outbound Communication (Tenant as WS Consumer)

The following figure illustrates the communication paths that have to be considered when using WS-Security.

The corresponding settings at tenant side are configured in the receiver SOAP adapter.

The table contains the security renewal use cases:

<table>
<thead>
<tr>
<th>Use Case</th>
<th>More information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tenant signs outbound request message.</td>
<td>Security Artifact Renewal for WS-Security (Tenant Signs Outbound Request) [page 179]</td>
</tr>
<tr>
<td>Tenant encrypts outbound request message.</td>
<td>Security Artifact Renewal for WS-Security (Tenant Encrypts Outbound Request) [page 180]</td>
</tr>
<tr>
<td>Tenant verifies outbound response message.</td>
<td>Security Artifact Renewal for WS-Security (Tenant Verifies Outbound Response) [page 182]</td>
</tr>
<tr>
<td>Tenant decrypts outbound response message.</td>
<td>Security Artifact Renewal for WS-Security (Tenant Decrypts Outbound Response) [page 183]</td>
</tr>
</tbody>
</table>

\[Additional notes on the figure and table content could be added here.\]
5.1.4.4.1 Security Artifact Renewal for WS-Security (Tenant Verifies Inbound Request)

This use case covers all situations where private keys used by a sender to sign messages sent to the tenant (in our terminology: inbound messages) are changed. The renewal process ensures that the related public verification key is changed on the tenant side so that no downtime is required.

**Note**

It depends on the [operating model](#) whether the tenant administrator and the integration developer are at the customer or at SAP. In the [customer-managed operating model](#), the tenant administrator and integration developer tasks are performed by the customer. In the [SAP-managed operating model](#), these tasks are performed by SAP.

The verifier (specified in the integration flow as using the WS-Security standard) uses a public key to verify a request payload signed by the sender. This public key has been imported into the tenant keystore as an X.509 certificate. The verifier uses the information provided in the WS-Security payload to determine the public key in the keystore.

The following figure illustrates the communication path for this use case.

![Communication Path Diagram](#)

1. **Sender administrator**: Creates new key pair/certificate.
2. **Sender administrator**: Provides the tenant administrator with the new certificate.
3. **Tenant administrator**: Adds the new certificate to the tenant keystore.
4. **Tenant administrator**: Informs the sender administrator that payloads signed with the new key can be sent.
5. **Sender administrator**: Configures sender system to send payloads signed with the new private key from now on.
6. **Sender administrator**: Removes the old key pair/certificate.
7. **Sender administrator**: Informs the tenant administrator that the sender system sends payloads signed with the new key.
8. **Tenant administrator**: Removes the old certificate from the keystore after a certain time period (which must be at least the guaranteed delivery time). This time period is necessary to make sure that payloads signed with the old key are no longer in the system.
The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

How WS-Security Works [page 226]
Involved Roles [page 135]

5.1.4.4.2 Security Artifact Renewal for WS-Security (Tenant Decrypts Inbound Request)

This use case covers all situations where private keys used by the tenant to decrypt request messages from a sender (in our terminology: inbound messages) are changed. The renewal process ensures that the related public encryption key is changed on the sender side so that no downtime is required.

The following figure illustrates the communication path for this use case.

![Communication Path Diagram]

Legend
- 🔒 Tenant private key
- 🔓 Tenant public key

i Note

It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

1. Tenant administrator: Creates new key pair/certificate.
2. Tenant administrator: Adds the new certificate to the keystore.
3. Tenant administrator: Provides the sender administrator with the new certificate and informs the sender administrator that payloads encrypted with the new key can be sent.
4. Sender administrator: Configures sender system to send payloads encrypted with the new public key/certificate from now on.
5. Sender administrator: Removes the old public key/certificate.
6. Sender administrator: Informs the tenant administrator that the sender system sends payloads encrypted with the new key.
7. Tenant administrator: Removes the old key pair/certificate from the keystore after a certain time period (which must be at least the guaranteed delivery time). This time period is necessary to make sure that payloads encrypted with the old key are no longer in the system.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

How WS-Security Works [page 226]
Involved Roles [page 135]

5.1.4.4.3 Security Artifact Renewal for WS-Security (Tenant Signs Inbound Response)

This use case covers all situations where private keys (used by the tenant to sign inbound response messages based on WS-Security) are changed. The renewal process ensures that the related public verification key is changed on the sender side (that receives the response message) so that no or only a minimum downtime is required.

The signer (if configured to use the WS-Security standard) uses a private key to sign a payload. The private key is provided in the tenant keystore. To locate the private key in the keystore, an alias is specified in the corresponding integration flow signer step.

The signed WS-Security data contains either the certificate of the public key corresponding to the private key or the issuer and serial version number of the certificate so that the receiver can easily determine the public key with which the signature must be verified.

The renewal process depends on whether the sender system (that sends the request message and receives the response message) can verify signed data with different public keys at the same time.

The following figure illustrates the communication path for this use case.
Note
It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

Sender is Able to Verify Payloads Signed by the Old Key and Payloads Signed by the New Key at the Same Time

1. Tenant administrator: Creates a new key pair.
2. Tenant administrator: Provides the new certificate to the sender administrator.
3. Sender administrator: Configures the sender system so that it is able to verify payloads signed by the old key and payloads signed by the new key.
4. Sender administrator: Informs the tenant administrator that the sender system is able to verify payloads signed by the old key and payloads signed by the new key.
5. Tenant administrator: Exchanges the old key pair with the new key pair, keeping the old alias. From now on, outbound messages are signed with the new key.
6. Tenant administrator: Informs the sender administrator that the key pair has been exchanged.
7. Sender administrator: Removes the old key pair. From now on, the sender system can only verify payloads signed by the new key.

Note
The process is similar to the WS-Security Signer Outbound Request case, however, the role of the receiver administrator is replaced by the role of the sender administrator, and the receiver system is replaced by the sender system.
Sender is Only Able to Verify Payloads Signed by the Same Key at One Time

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Provides the new certificate to the sender administrator.
3. Tenant administrator: Agrees a downtime with the sender administrator so that no signed messages are sent during the certificate exchange.
4. Sender administrator: Stops sending signed messages at the start of the agreed downtime.
5. Sender administrator: Exchanges the old certificate with the new certificate.
6. Sender administrator: Informs the tenant administrator that message sending has been stopped and that the certificate has been exchanged.
7. Tenant administrator: Ensures that there are no messages signed with the old key in the system.
8. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate, keeping the old alias in the keystore.
   From now on, data is signed with the new key.
9. Tenant administrator: Informs the sender administrator that the key pair/certificate has been exchanged.
10. Sender administrator: Starts sending messages.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

How WS-Security Works [page 226]
Involved Roles [page 135]

5.1.4.4.4 Security Artifact Renewal for WS-Security (Tenant Encrypts Inbound Response)

This use case covers all situations where a private key used to decrypt a response message (of an inbound request) is changed on the side of the sender of the response. The renewal process ensures that the related public encryption key is changed on the tenant side so that no or minimum downtime is required.

The WS-Security encryptor uses a public key to encrypt the payload of the inbound response message.

The following figure illustrates the communication path for this use case.
Note
It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

Sender is Able to Decrypt Payloads Encrypted by the Old Key and Payloads Encrypted by the New Key at the Same Time

1. Sender administrator: Creates a new key pair/certificate.
2. Sender administrator: Configures the sender system so that it can decrypt payloads encrypted by the old key and payloads encrypted by the new key.
3. Sender administrator: Provides the tenant administrator with the new certificate and informs the tenant administrator that the sender system can verify payloads signed by the old key and payloads signed by the new key.
4. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate in the keystore, keeping the old alias.
   From now on, data is encrypted with the new key.
5. Tenant administrator: Informs the sender administrator that the certificate has been exchanged.
6. Sender administrator: Removes the old certificate so that from now on the sender system can only verify payloads signed by the new key.

Sender is Only Able to Decrypt Payloads Encrypted by the Same Key at One Time

1. Sender administrator: Creates a new key pair/certificate.
2. Tenant administrator: Agrees a downtime with the sender administrator so that no encrypted messages are sent during the certificate exchange.
3. Sender administrator: Stops sending messages at the start of the agreed downtime.
4. Sender administrator: Exchanges the old key pair/certificate with the new key pair/certificate.
5. Sender administrator: Provides the tenant administrator with the new certificate and informs the tenant administrator that message sending has been stopped and that the key pair/certificate has been exchanged.
6. Tenant administrator: Ensures that there are no messages encrypted with the old key in the system.
7. Tenant administrator: Exchanges the old certificate with the new certificate, keeping the old alias in the keystore. From now on, data is encrypted with the new key.
8. Tenant administrator: Informs the sender administrator that the key pair/certificate has been exchanged.
9. Sender administrator: Starts sending messages.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

**Related Information**

How WS-Security Works [page 226]
Involved Roles [page 135]

**5.1.4.4.5 Security Artifact Renewal for WS-Security (Tenant Signs Outbound Request)**

This use case covers all situations where private keys (used by the tenant to sign outbound messages based on WS-Security) are changed. The renewal process ensures that the related public verification key is changed on the receiver side so that no or only a minimum downtime is required.

The signer (if configured to use the WS-Security standard) uses a private key to sign a payload. The private key is provided in the outbound keystore of the tenant. To locate the private key in the keystore, an alias is specified in the corresponding integration flow signer step.

The signed WS-Security data contains either the certificate of the public key corresponding to the private key or the issuer and serial version number of the certificate so that the receiver can easily determine the public key with which the signature must be verified.

The renewal process depends on whether the receiver system can verify signed data with different public keys at the same time.

**Receiver is Able to Verify Payloads Signed by the Old Key and Payloads Signed by the New Key at the Same Time**

The same process applies as for the PKCS#7/CMS Signer in the same case.
Receiver is Only Able to Verify Payloads Signed by the Same Key at One Time

The same process applies as for the XML Digital Signer in the same case.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

How WS-Security Works [page 226]
Renewal of Keys for CMS/PKCS#7 Signer - Outbound [page 151]
Renewal of Keys for XML Digital Signature Signer - Outbound [page 168]

5.1.4.4.6 Security Artifact Renewal for WS-Security (Tenant Encrypts Outbound Request)

This use case covers all situations where a private key used for message decryption is changed by a receiver. The renewal process ensures that the related public encryption key is changed on the tenant side so that no or minimum downtime is required.

The encryptor uses a public key to encrypt a payload.

This public key is provided in the tenant keystore with an X.509 certificate. The encryptor uses an alias configured in the corresponding integration flow step to locate the public key in the keystore. The encrypted WS-Security data contains either the certificate of the public key corresponding to the private key or the issuer and serial version number of the certificate so that the receiver can easily determine the certificate and private key to be used for the decryption.

The renewal process depends on whether the receiver system can decrypt XML Encryption data with different public keys at the same time.

The following figure illustrates the communication path for this use case.
Receiver is Able to Verify Payloads Encrypted by the Old Key and Payloads Encrypted by the New Key at the Same Time

1. Receiver administrator: Creates a new key pair/certificate.
2. Receiver administrator: Configures the receiver system so that it can decrypt payloads encrypted by the old key and payloads encrypted by the new key.
3. Receiver administrator: Provides the tenant administrator with the new certificate and informs the tenant administrator that the receiver system can verify payloads signed by the old key and payloads signed by the new key.
4. Tenant administrator: Exchanges the old key pair/certificate with the new key pair/certificate in the keystore, keeping the old alias.
   From now on, data is encrypted with the new key.
5. Tenant administrator: Informs the receiver administrator that the certificate has been exchanged.
6. Receiver administrator: Removes the old certificate so that from now on the receiver system can only verify payloads signed by the new key.

Receiver is Only Able to Verify Payloads Encrypted by the Same Key at One Time

1. Receiver administrator: Creates a new key pair/certificate.
2. Receiver administrator: Provides the tenant administrator with the new certificate.
3. Tenant administrator: Agrees a downtime with the receiver administrator so that no encrypted messages are sent during the certificate exchange.
4. Tenant administrator: Informs the integration developer that the integration flow that specifies the encryption of the payload must be undeployed.
   From now on, no encrypted messages are sent to the receiver system.
6. Integration developer: Informs the tenant administrator about the preceding step.
7. Tenant administrator: Exchanges the old certificate with the new certificate, keeping the old alias in the keystore.
   From now on, data is encrypted with the new key.
8. Tenant administrator: Informs the receiver administrator that no encrypted messages are being sent.
9. Receiver administrator: Exchanges the key pair/certificate in the receiver system.
10. Receiver administrator: Informs the tenant administrator that from now on the receiver system expects payloads encrypted by the new key.
11. Tenant administrator: Informs the integration developer that the integration flow can be redeployed.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

Related Information

How WS-Security Works [page 226]
Involved Roles [page 135]

5.1.4.4.7 Security Artifact Renewal for WS-Security (Tenant Verifies Outbound Response)

This use case covers all situations where private keys used by a sender to sign response messages sent to the tenant are changed.

The WS-Security verifier uses a public key to verify a WS-Security response payload.

The following figure illustrates the communication path for this use case.

Legend
- Receiver private key
- Receiver public key

i Note

It depends on the operating model whether the tenant administrator and the integration developer are at the customer or at SAP. In the customer-managed operating model, the tenant administrator and integration developer tasks are performed by the customer. In the SAP-managed operating model, these tasks are performed by SAP.

1. Receiver administrator: Creates a new key pair/certificate.
2. Receiver administrator: Provides the tenant administrator with the new certificate.
3. Tenant administrator: Adds the new certificate to the tenant keystore.
4. Tenant administrator: Informs the receiver administrator that payloads signed with the new key can be sent.
5. Receiver administrator: Configures receiver system to send payloads signed with the new private key from now on.
6. Receiver administrator: Removes the old key pair/certificate.
7. Receiver administrator: Informs the tenant administrator that the receiver system sends payloads signed with the new key.
8. Tenant administrator: Removes the old certificate from the keystore after a certain time period (which must be at least the guaranteed delivery time). This time period is necessary to make sure that payloads signed with the old key are no longer in the system.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

**Related Information**

How WS-Security Works [page 226]
Involved Roles [page 135]

### 5.1.4.4.8 Security Artifact Renewal for WS-Security (Tenant Decrypts Outbound Response)

This use case covers all situations where private keys used by the tenant to decrypt a response message (from an outbound request) are changed. The renewal process ensures that the related public encryption key is changed on the side of the receiver of the request (that sends the response) so that no downtime is required.

The WS-Security decryptor uses a private key to verify a WS-Security response payload.

The following figure illustrates the communication path for this use case.
It depends on the **operating model** whether the tenant administrator and the integration developer are at the customer or at SAP. In the **customer-managed operating model**, the tenant administrator and integration developer tasks are performed by the customer. In the **SAP-managed operating model**, these tasks are performed by SAP.

1. Tenant administrator: Creates a new key pair/certificate.
2. Tenant administrator: Adds the new certificate to the keystore.
3. Tenant administrator: Provides the receiver administrator with the new certificate and informs the receiver administrator that payloads encrypted with the new key can be sent.
4. Receiver administrator: Configures receiver system to send payloads encrypted with the new public key/certificate from now on.
5. Receiver administrator: Removes the old public key/certificate.
6. Receiver administrator: Informs the tenant administrator that the receiver system sends payloads encrypted with the new key.
7. Tenant administrator: Removes the old key pair/certificate from the keystore after a certain time period (which must be at least the guaranteed delivery time).
   This time period is necessary to make sure that payloads encrypted with the old key are no longer in the system.

The participants have to make sure that old keys are kept for at least 90 days after they have been exchanged for new ones. This is to ensure that messages can still be decrypted with the old keys for a period of time.

**Related Information**

- How WS-Security Works [page 226]
- Involved Roles [page 135]

**5.2 Renewal of Keys Provided by SAP**

To enable secure communication between the tenant and connected remote systems, the system keystore deployed on the tenant must contain up-to-date keys owned by the tenant administrator and SAP.

The certificate management features of the Web UI support the key renewal process described below.

**⚠️ Caution**

You must exercise caution when renewing keys in order to avoid unplanned downtimes of your integration scenarios.

The tenant administrator has to orchestrate the key renewal process together with the administrators of all involved sender and receiver systems. It may be necessary, for example, to agree on a specific downtime in certain cases.

Note that it is the tenant administrator who should coordinate the whole process, as he or she knows all the administrators of the connected sender and receiver systems.
Downtime can be avoided in certain cases. In order to schedule and set up a key renewal process correctly, see the topics linked below.

We provide an example of a renewal process in a separate topic.

1. SAP prepares the new key pair 90 days before a key expires. The new key pair is imported into the customer tenant by SAP. The new key pair has the same alias as the key pair that is due to expire.
2. The tenant administrator can access the content of this keystore (Web UI Monitoring under Manage Keystore > New SAP Keys).
3. SAP informs the tenant administrator about the new key.
4. Shortly before the key expires, the tenant administrator downloads the X.509 certificate and certificate chain of the new key pair from the New SAP Keys keystore (Web UI Monitoring under Manage Keystore > New SAP Keys).
5. The tenant administrator informs the administrators of the connected sender and receiver systems that keys need to be renewed. If required, the administrators of the sender and receiver systems and the tenant administrator agree on a downtime during which the affected keys can be exchanged both in the sender and receiver systems and on the tenant.
6. The administrators of the sender and receiver systems update the keystores of their systems using the new certificate and certificate chain (provided to them by the tenant administrator). The tenant administrator activates the new key pair on the tenant with the certificate management features of the Web UI.

**Related Information**

- Keys Provided by SAP [page 185]
- Activating a New SAP Key Pair on the Tenant [page 186]
- Example: Renewal of Key Pairs Provided by SAP (Avoiding Downtime) [page 187]
- Renewal of SAP Keys Without Any Downtime [page 188]

### 5.2.1 Keys Provided by SAP

When SAP first provides a tenant to a customer, it delivers an X.509 certificate chain, which has to be renewed every 2 years.

SAP key pairs delivered with the tenant have the following aliases: `sap_cloudintegrationcertificate`, `hcicertificate`, or `hcicertificate1`.

You can use them in the following channels and integration flow steps:

- HTTPS outbound connections with client-certificate authentication
- Signature creator steps (XML Signature, CMS/PKCS#7, WS Security in SOAP adapter)
- Decryptor steps (CMS/PKCS#7, WS Security in SOAP adapter)
In all use cases, the corresponding X.509 certificate has to be made available to the sender or receiver system to enable the following steps:

- Configuring the required certificate-user mapping (HTTPS communication)
- Verifying the signature
- Encrypting the message

If the key pair is renewed without exchanging the certificate with the receiver or sender system, message processing will fail.

### 5.2.2 Activating a New SAP Key Pair on the Tenant

SAP Cloud Platform Integration comes with a set of features that facilitate the tenant administrator’s task of renewing keys provided by SAP on the tenant.

The following different keystore types enable the tenant administrator to renew keys in an efficient way:

- **System keystore**
  This keystore is located in the `system.jks` file provided with the tenant and contains all keys that are actively used by the deployed integration flows.

- **New SAP Keys keystore**
  This keystore contains keys already prepared by SAP before a key pair expires.
  As the key expiry date approaches, the tenant administrator can do the following:
  - Download the new SAP keys from the KeyRenewal keystore to share them with the administrators of the connected back ends
  - Activate the keys relevant for the tenant

- **SAP Key History keystore**
  This keystore contains old (expired) keys.
  If required, the tenant administrator can restore a key from the Key History keystore and use it to replace a key in the system keystore.

During the activation of an SAP key, the system performs three steps, as shown in the following figure.
1. The old key pair (which expires soon) is added by the system to the Key History keystore.
2. The old key pair (in the system.jks keystore) is overwritten by the new key pair from the New SAP Keys keystore.
3. In the New SAP Keys keystore, the new key pair is removed.

**Note**
To restore an old key pair, the tenant administrator can copy the key pair from the SAP Key History keystore to the New Keys keystore and activate it there.

### 5.2.3 Example: Renewal of Key Pairs Provided by SAP (Avoiding Downtime)

In the following example, an SAP key pair is used in the following steps and channels (in several integration flows):
- CMS signer
- CMS decryptor
- HTTPS outbound channel

In addition, the sender and receiver systems meet the requirement that they can cope with two certificates at the same time.

The tenant administrator refers to the general key renewal documentation and finds out that key renewal can be accomplished without any downtime if the following conditions apply:
- All sender systems encrypting the message are capable of sending CMS messages encrypted by several public keys.
- All receiver systems verifying the CMS signature are capable of verifying CMS signatures signed with several private keys.
• All receiver systems called by the outbound HTTPS channels can cope with different client certificates coming from the same tenant (by using multiple certificate-user mappings).

To prepare for the renewal process, the tenant administrator asks all the administrators of the sender and receiver systems whether their systems meet these requirements. If the answer is yes, the keys can be renewed by following three major steps:

1. The tenant administrator provides all administrators of the sender and receiver systems with the X.509 certificate and the X.509 certificate chain of the new key pair. The administrators of the sender and receiver systems install these certificates in their systems so that the following conditions are met:
   - The CMS decryptor decrypts the CMS message with the old and new private key (the symmetric key is actually encrypted twice with the old and new private key).
   - The CMS verifier can verify CMS signatures signed by the old or new public key.
   - The HTTPS server can cope with the old and new client certificate.

2. The tenant administrator overwrites the old SAP key pair (in the tenant’s system keystore) with the new SAP key pair by activating the new SAP key pair.

3. The tenant administrator informs the administrators of the sender and receiver systems that the key has been renewed.
   The administrators then remove the old X.509 certificate from their system.

This process ensures that no downtime is necessary in message processing during key renewal.

→ Tip

You can see from this example that the tenant administrator has to have a good knowledge of the integration flow steps and channels where the key pair is used. Furthermore, the administrators of the sender and receiver systems have to know the capabilities of their systems.

If a sender/receiver administrator doesn't know whether the CMS decryptor component of his or her system is capable of creating a CMS Enveloped Data message where the symmetric key can be decrypted by two public keys, to be on the safe side we recommend that the sender and receiver system administrators and the tenant administrator agree on a downtime. During the downtime window, all involved parties adapt the configuration to the new key pair. Note that this downtime window can only be organized if the tenant administrator knows the administrators of the sender and receiver systems.

5.2.4 Renewal of SAP Keys Without Any Downtime

In certain cases it is possible to avoid a downtime in message processing when renewing keys. See the following use-case descriptions:

Renewal of Keys for CMS/PKCS#7 Signer - Outbound [page 151]
Renewal of Keys for CMS/PKCS#7 Decryptor - Inbound [page 159]
Renewal of OpenPGP Signer Key - Outbound [page 165]
Renewal of OpenPGP Encryption Key - Inbound [page 164]
Renewal of Keys for XML Digital Signature Signer - Outbound [page 168]
Security Artifact Renewal for WS-Security (Tenant Decrypts Inbound Request) [page 174]
These topics describe separate processes for renewing the various keys that are involved in an integration scenario using SAP Cloud Platform Integration. The topics cover situations where a key is owned either by the tenant administrator or by the administrator of the sender/receiver system connected to the tenant.

To keep things simple, the individual topics describe idealized situations where a key pair is used in a single step or only in one communication channel. In real-life situations, however, a key pair is typically used in several integration flows, integration flow steps, and communication channels. The tenant administrator needs to know all the steps and channels where the key pair is used to be able to correctly define the key renewal process.
6 Support Tasks

When the customer-managed operating model is applied, the tenant cluster is managed by the customer. If support activities have to be performed on the tenant cluster by an SAP expert, the following steps are required.

As tenant administrator (at customer side) perform the following steps:

1. Ask your SAP contact to entitle an SAP expert as system development expert.
   SAP provides you with the corresponding user ID of the dedicated SAP expert.
2. Temporarily (for the time the support activities are going on) assign the authorization group `AuthGroup.SystemDeveloper` to this user.

Provide the tenant administrator (at customer side) with the user ID of the SAP expert and ask him or her to temporarily (for the time the support activities are going on) assign the authorization group `AuthGroup.SystemDeveloper` to this user.
7 Additional Features

7.1 Health checks and recommended actions for SAP Integration Advisor Node

Currently, the following health checks are covered for Integration Advisor node:

<table>
<thead>
<tr>
<th>Health Check</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Advisor State</td>
<td>The MBean checks the health of Integration Advisor. The threshold check is String 'Available'. In case Integration Advisor’s health is good the check returns ‘Available’ and ‘Unavailable’ in case Integration Advisor’s health is poor.</td>
</tr>
<tr>
<td>Availability Check</td>
<td>It checks the availability of the application via HTTP ping. The check is executed every second. Default values of 50 seconds for warning and 60 seconds for critical are currently assigned to this check. This means that if the ping response time is more than 50 or 60 seconds a warning or critical alert will be generated respectively.</td>
</tr>
</tbody>
</table>

7.2 Enabling SAP Solution Manager to Act as Additional Alert Consumer

You can enable SAP Solution Manager to display alerts.

Context

Currently, alerts are supported that are raised if no receiver can be determined during routing.

To set up this scenario, the SaaS administrator and the administrators of the involved SAP Solution Manager systems have to perform the following steps.

i Note

For more information on the tasks of the administrator of the SAP Solution Manager systems, see the documentation for SAP Solution Manager.
**Procedure**

1. Provide the administrator of the SAP Solution Manager system with the tenant management node URL. The tenant management node URL is contained in the email that is sent out to the customer after the tenant provisioning process.
2. Create a user that enables the SAP Solution Manager system to connect to the integration platform as a client.
3. Assign the following role to this user: `IntegrationOperationServer.consumealerts`.
4. Provide the administrator of the SAP Solution Manager system with this user.
5. Make sure that the SAP Solution Manager system is registered as an alert consumer.
   - To request this feature, create a ticket on component `LOD-HCI-PI-OPS`.

**Results**

Once the connected SAP Solution Manager systems and tenants have been configured accordingly, alerts of the above-mentioned type are displayed in the Exception Management inbox of the involved SAP Solution Manager systems.
8 Concepts of Secure Communication

There are several options to protect the message exchange. You can secure the communication on transport level by selecting the HTTPS or SFTP protocol and installing specific authentication methods. In addition to that, you can set up methods to encrypt and decrypt the content of the message and to digitally sign and verify the message.

Related Information

Basics [page 193]
Security Elements [page 238]

8.1 Basics

Related Information

HTTPS-Based Communication [page 193]
SFTP-Based Communication [page 216]
Message-Level Security [page 220]
Certificate Management [page 232]

8.1.1 HTTPS-Based Communication

Related Information

Authentication and Authorization Options (Inbound) [page 194]
Authentication Options (Outbound) [page 203]
Load Balancer Root Certificates Supported by SAP [page 214]
8.1.1.1 Authentication and Authorization Options (Inbound)

When a client calls a server using a secure communication channel, two different kinds of checks are performed subsequently.

- **Authentication**
  Verifies the identity of the calling entity.

- **Authorization**
  Checks what a user or other entity is authorized to do (for example, as defined by roles assigned to it). In other words, the authorization check evaluates the access rights of a user or other entity.

When a client calls a server, it is first authenticated and, in a subsequent step, the authorization check is performed.

We use **inbound** to refer to the communication direction when a sender system sends a message to the integration platform.

## Combinations of Authentication and Authorization (Inbound)

For inbound communication based on HTTPS, the authentication and authorization options can be combined in a specific way.

### Combination of Authentication/Authorization Options

<table>
<thead>
<tr>
<th>Authentication Option</th>
<th>Can Be Used with the Following Authorization Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic authentication</td>
<td>Role-based authorization</td>
</tr>
<tr>
<td>The sender (client) authenticates itself against the server based on user credentials (user name and password). The HTTP header of the inbound message (from the sender) contains the user name and password.</td>
<td>For this user, the authorizations are checked based on user-to-role assignments defined on the tenant.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Client-certificate authentication and certificate-to-user mapping</th>
<th>Role-based authorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>The sender (client) authenticates itself against the server based on a digital client certificate. Furthermore, this certificate is mapped to a user (based on the information contained in a Certificate-to-User Mapping artifact deployed on the tenant).</td>
<td>For the user derived from the certificate-to-user mapping, the authorizations are checked based on user-to-role assignments defined on the tenant.</td>
</tr>
</tbody>
</table>

### Note

You can map multiple certificates to the same user (n:1 certificate-to-user mappings possible).
Authentication Options (Inbound)

For inbound communication, different ways are supported how the sender can authenticate itself against Cloud Integration.

We use **inbound** to refer to the communication direction when a sender system sends a message to the integration platform.

- **Basic authentication**
  The calling entity is authenticated based on credentials (user name and password)

- **Client-certificate authentication and certificate-to-user mapping**
  The calling entity is authenticated based on a certificate, and the certificate is mapped to a user (for which the authorization check is executed in a subsequent step).

More information: [OAuth 2.0 Specification](#)
- **Client-certificate authentication** (without certificate-to-user mapping)
- OAuth 2.0
  OAuth allows you to set up authentication scenarios without the need to share credentials.
  More information on the concepts:
  - Protecting Applications with OAuth 2.0
  - OAuth 2.0 Specification

**Related Information**

- Basic Authentication [page 200]
- Client Certificate Authentication and Certificate-to-User Mapping (Inbound) [page 196]
- Client Certificate Authentication (Inbound) [page 198]

### 8.1.1.1.1 Client Certificate Authentication and Certificate-to-User Mapping (Inbound)

This option includes an authentication step based on a digital client certificate and the mapping of the certificate to a user.

With a certificate-to-user mapping, a certificate is mapped to a user, and that way the user can be authenticated based on a certificate.

**i Note**

Note that multiple certificates can be mapped to one user (n:1 certificate-to-user mappings possible).

Certificate-to-user mappings make sure that a user is always associated with the certificate as a whole, not only with one attribute of it (for example the common name (CN)). As different certificates can have the same CN, mapping only the CN to a user name bears the risk that different certificates can be mapped accidentally to the same user. Using certificate-to-user mappings circumvents this risk.

For the user defined that way, in a subsequent step, an authorization step is being executed.

**How it Works**

The following figure shows the complete setup of components and security artifacts required for this option.
When you have configured this authentication option, the authentication of the user is performed in the following way at runtime:

The TLS connection of the sender system and the integration platform is terminated and newly established by the load balancer. This means, that first the load balancer authenticates itself against (as server) the sender based on the load balancer server certificate. Vice versa, the sender authenticates itself against the load balancer as client using the sender client certificate.

To enable the sender to communicate that way with the load balancer, the sender administrator has to make sure that the sender client certificate is signed by one of the certification authorities that are supported by the load balancer.

The load balancer sets the following message header fields:

- **SSL_CLIENT_CERT**
  Contains the Base64-encoded sender client certificate.

- **SSL_CLIENT_USER**

When the authentication is been executed successfully, the load balancer writes the sender client certificate (base 64-encoded) into the message header (field SSL_CLIENT_CERT). The tenant then maps the sender client certificate to a user based on the certificate-to-user mapping which is deployed on the tenant.

**Note**

In a subsequent step, the authorization check is executed for the default role (ESBMessaging.send) provided by SAP or a custom role configured in an adapter. You can define a custom role using Cloud Platform Role Management. For more information, read the blog on [How to Setup Secure HTTP Inbound Connection with Client Certificates](https://example.com).
8.1.1.1.2 Client Certificate Authentication (Inbound)

This option includes an authentication step based on a digital client certificate.

How it Works

The following figure shows the complete setup of components and security artifacts required for this option.

![Diagram showing setup of components and security artifacts]

When you have configured this authentication option, the authentication of the user is performed in the following way at runtime:

The TLS connection of the sender system and the integration platform is terminated and newly established by the load balancer. This means, that first the load balancer authenticates itself against (as server) the sender based on the load balancer server certificate. Vice versa, the sender authenticates itself against the load balancer as client using the sender client certificate.

To enable the sender to communicate that way with the load balancer, the sender administrator has to make sure that the sender client certificate is signed by one of the certification authorities that are supported by the load balancer.

The load balancer sets the following message header fields:

- **SSL_CLIENT_CERT**
  Contains the Base64-encoded sender client certificate.
- **SSL_CLIENT_USER**
## Required Security Material

### Certificates for Inbound Message Processing

<table>
<thead>
<tr>
<th>Keystore</th>
<th>Certificate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender keystore</td>
<td>Load balancer server root certificate (identifies CA that has signed the load balancer server certificate)</td>
<td>This certificate is required to identify the root CA at the top of the certificate chain that ultimately guarantees the trustability of the load balancer server certificate. In many cases, there is a multilevel setup of CAs so that a certificate is signed by an intermediate CA. The trustability of the intermediate CA is guaranteed by another intermediate CA one level higher, and so on, up to the root CA at the top of the <strong>certificate chain</strong>. In this case, it is necessary to assign the certificate chain to the certificate, to enable the connected component (which has imported only the root CA into its keystore) to evaluate the chain of trust.</td>
</tr>
<tr>
<td>Sender client certificate</td>
<td>This certificate is required to authenticate the sender (client) when calling Cloud Integration. On the Cloud Integration tenant side, this certificate is required to configure the authorization check.</td>
<td></td>
</tr>
<tr>
<td>Load balancer keystore</td>
<td>Load balancer server certificate</td>
<td>This certificate is required to identify the load balancer as a trusted server (to which clients like the sender system can connect).</td>
</tr>
<tr>
<td>Sender client root certificate</td>
<td>This certificate is required to identify the root CA at the top of the certificate chain that ultimately guarantees the trustability of the sender client certificate. There is a list of CAs that are supported by the load balancer.</td>
<td><strong>More information:</strong> <a href="#">Load Balancer Root Certificates Supported by SAP</a></td>
</tr>
</tbody>
</table>

For sakes of completeness, note that always a tenant keystore (not depicted in the figure) needs to be available to enable the system to do an additional outbound communication step that is required for technical purposes: The basic technical connectivity of a cluster is checked on a regular basis, as soon as the cluster is active. For
this purpose, every 30 seconds the tenant management node sends an HTTPS request to an assigned runtime node via the load balancer. This simulates an external call to the runtime node. To enable this communication, a keystore needs to be deployed on the tenant, containing a valid client certificate that is accepted by the load balancer as well as the root certificate of the same. If this keystore is not available or contains an invalid certificate, the cluster will raise an error. The keystore and required certificate are provisioned by SAP together with the tenant.

**Note**

In a subsequent authorization check, the permissions of the sender are checked on the tenant by evaluating the distinguished name (DN) of the client certificate of the sender. The client certificate of the sender is being passed through to the tenant by the load balancer (in the message header). To provide the tenant with the information on the correct client certificate to be expected from the sender, a corresponding setting has to be made in the related integration flow.

### 8.1.1.1.3 Basic Authentication

Basic authentication allows a client to authenticate itself against the server based on user credentials (user name and password).

**Caution**

Consider that we do not recommend to use basic authentication in productive scenarios because of the following security aspects:

Basic authentication has the risk that authentication credentials, for example, passwords, are sent in clear text. Using TLS (transport-layer security, also referred to as Secure Sockets Layer) as transport-level encryption method (when using HTTPS as protocol) makes sure that this information is nevertheless encrypted on the transport path. However, the authentication credentials might become visible to SAP-internal administrators at points in the network where the TLS connection is terminated, for example, load balancers. If logging is not done properly at such devices, the authentication credentials might become part of log files. Also network monitoring tools used at such devices might expose the authentication information to administrators. Furthermore, the person to whom the authentication credentials belong (in the example above, the password owner) needs to maintain the password in a secure place.

**How it Works**

The following figure shows the setup of components required for inbound basic authentication.
These are the steps at runtime:

The HTTP header of the inbound message (from the sender) contains user name and password. To protect these credentials during the communication step, the connection is secured using TLS (SSL).

This includes a step where the load balancer authenticates itself as server against the sender based on a certificate. To enable this security measure, the keystore of the load balancer contains a server certificate signed by a certification authority. To be more precise, the keystore of the load balancer contains a complete certificate chain from (including all intermediate certificates). On the other side of the communication, the keystore of the connected sender system must contain the load balancer server root certificate. That is the certificate that identifies the certification authority (CA) that signed the load balancer’s server certificate (on top of the certificate chain).

The other way round, the identity of the sender is checked by SAP evaluating the credentials (user and password) against the user.

It is also depicted in the figure that the authentication option needs to be activated for the corresponding integration flow.

**Required Security Material**

To enable the sender system to authenticate itself against the integration platform with basic authentication, a communication user has to be created for the sender.

The following figure provides an overview of the involved security artifacts and storage locations.
## Certificates for Inbound Message Processing

<table>
<thead>
<tr>
<th>Keystore</th>
<th>Certificate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender keystore</td>
<td>Load balancer server root certificate</td>
<td>This certificate is required to identify the root CA at the top of the certificate chain that ultimately guarantees the trustability of the load balancer server certificate. In many cases, there is a multilevel setup of CAs so that a certificate is signed by an intermediate CA. The trustability of the intermediate CA is guaranteed by another intermediate CA one level higher, and so on, up to the root CA at the top of the certificate chain. In this case, it is necessary to assign the certificate chain to the certificate, to enable the connected component (which has imported only the root CA into its keystore) to evaluate the chain of trust.</td>
</tr>
<tr>
<td>Load balancer keystore</td>
<td>Load balancer server certificate</td>
<td>This certificate is required to identify the load balancer as a trusted server (to which clients like the sender system can connect).</td>
</tr>
</tbody>
</table>

For sakes of completeness, note that always a tenant keystore (not depicted in the figure) needs to be available to enable the system to do an additional outbound communication step that is required for technical purposes: The basic technical connectivity of a cluster is checked on a regular basis, as soon as the cluster is active. For this purpose, every 30 seconds the tenant management node sends an HTTPS request to an assigned runtime node via the load balancer. This simulates an external call to the runtime node. To enable this communication, a keystore needs to be deployed on the tenant, containing a valid client certificate that is accepted by the load balancer as well as the root certificate of the same. If this keystore is not available or contains an invalid certificate, the cluster will raise an error. The keystore and required certificate are provisioned by SAP together with the tenant.

### 8.1.1.1.2 Authorization Options (Inbound)

For inbound HTTPS requests, two different ways to check the authorization of the caller can be configured.

We use **inbound** to refer to the communication direction when a sender system sends a message to the integration platform.

- **Role-based authorization**
  
  The permissions of the calling entity (user) are checked based on a user-to-role assignments configured in the associated identity provider.

  In the related sender adapter, you can assign the role based on which the inbound authorization is to be checked for the integration flow.
Subject/Issuer DN authorization check
The distinguished name (DN) of a certificate (associated with the calling entity) is checked. Subject/Issuer DN authorization check can be defined for individual integration flows.

Related Information
Role-Based Authorization [page 203]
Subject/Issuer DN authorization check [page 203]

8.1.1.2.1 Role-Based Authorization
This option allows you to define permissions for users in the connected identity provider (by default, SAP Identity Service) and to perform an authorization check based on these settings.

For HTTPS requests sent to Cloud Integration, it is checked if the role `ESBMessaging.send` is assigned to the user.

The permissions of the sending client are checked according to roles assigned to the user in the associated identity provider.

User management (which includes the assignment of permissions to users) is performed by the tenant administrator using the SAP Cloud Platform Cockpit.

8.1.1.2.2 Subject/Issuer DN authorization check
It is checked (for a specific integration flow) if the subject/issuer distinguished name (DN) of the assigned certificate matches the incoming certificate.

If yes, this specific integration flow can be processed. The authorization check is performed based on the distinguished name (DN) of the client certificate. The DN has to be specified when configuring the relevant integration flow.

8.1.1.2 Authentication Options (Outbound)
For outbound communication through HTTPS (when the tenant sends a message to a receiver), the following authentication options are supported.

- **Basic authentication**
  The calling entity (tenant) is authenticated based on credentials (user name and password)

- **Client-certificate authentication**
  The calling entity (tenant) is authenticated based on a certificate.

- **OAuth** (when using the Twitter or Facebook receiver adapter)
8.1.1.2.1 Basic Authentication

Basic authentication allows a the tenant to authenticate itself against the receiver through credentials (user name and password).

How it Works

The following figure shows the setup of components required for this authentication option.

```
<table>
<thead>
<tr>
<th>Tenant</th>
<th>Request</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keystore</td>
<td>User Credentials Artefacts</td>
<td>Keystore</td>
</tr>
<tr>
<td>Receiver server root certificate</td>
<td>User credentials (username, password)</td>
<td>Receiver server certificate (contains certificate chain)</td>
</tr>
<tr>
<td>Database</td>
<td></td>
<td>user, roles</td>
</tr>
</tbody>
</table>
```

Basic authentication for HTTPS-based outbound calls works the following way:

1. The tenant (client) sends a message to the customer back-end system. The HTTP header of the message contains user credentials (name and password).
   To protect the user credentials during the communication step, the connection is secured using SSL.
2. The customer back-end authenticates itself as server against the tenant using a certificate (the customer back-end identifies itself as trusted server).
   To support this, the keystore of the customer back-end system must contain a server certificate signed by a certification authority. To be more precise, the keystore must contain the complete certificate chain. On the other side of the communication, the keystore of the connected tenant must contain the customer back-end server root certificate.
3. The tenant is authenticated by the customer back-end by evaluating the credentials against the user stored in a related data base connected to the customer back-end.
Required Security Material

Certificates for Outbound Message Processing

<table>
<thead>
<tr>
<th>Keystore</th>
<th>Security Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keystore (tenant-specific)</td>
<td>Receiver server root certificate</td>
<td>This certificate is required to identify the root CA that is at the top of the certificate chain that ultimately guarantees the trustability of the receiver server certificate.</td>
</tr>
<tr>
<td></td>
<td>Receiver server certificate (signed by CA with which the tenant has a trust relationship)</td>
<td>This certificate is required to identify the receiver (to which the tenant connects as the client) as a trusted server.</td>
</tr>
<tr>
<td>Receiver keystore</td>
<td>Receiver server root certificate</td>
<td></td>
</tr>
<tr>
<td>User credentials artifact</td>
<td>User and password</td>
<td>With these credentials the tenant authenticates itself as client at the receiver system.</td>
</tr>
</tbody>
</table>

8.1.1.2.2 Client Certificate Authentication (Outbound)

The following figure shows the setup of components required for this authentication option.

How it Works

HTTPS client certificate authentication
The tenant authenticates itself against the receiver based on a certificate.

This authentication option works the following way:

1. The tenant sends a message to the receiver.
2. The receiver authenticates itself (as trusted server) against the tenant when the connection is being set up. In this case, the receiver acts as server and the authentication is based on certificates.
3. Authentication of the tenant: The identity of the tenant is checked by the receiver by evaluating the client certificate chain of the tenant. As prerequisite for this authentication process, the client root certificate of the tenant has to be imported into the receiver keystore (prior to the connection set up). As CA who provides the root certificate, Cyber trust Public Sure Server SV CA is used. Steps 2 and 3 are referred to as mutual SSL handshake.
4. Authorization check: The permissions of the client (tenant) are checked in a subsequent step by the receiver.

### 8.1.1.2.3 OAuth 2.0

OAuth 2.0 allows a user to grant a client access to a protected resource (hosted by a resource server). The user typically restricts the access of the client and doesn’t allow full access.

OAuth 2.0 (Open Authorization) is an open standard for authorization. It enables users, for example the owners of a protected source, to grant clients restricted access (scope) to their data, that is, the protected source without sharing their authorization details. This data is hosted by a resource server (in terms of Cloud Integration outbound communication, the receiver system). This means, users restrict access and keep credentials private. In Cloud Integration, the Twitter, Facebook, http, Mail, OData, SFSF, and AMQP adapters support the OAuth 2.0 authorization standard.

OAuth 2.0 uses the following components:

<table>
<thead>
<tr>
<th>Component/Term</th>
<th>Description</th>
<th>In the context of Cloud Integration outbound communication, this is ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource owner</td>
<td>Owns the data and allows access to it.</td>
<td>Administrator of the receiver system</td>
</tr>
<tr>
<td>Resource server</td>
<td>Hosts data and accounts of the resource owner.</td>
<td>Receiver system (functions of the receiver system called by Cloud Integration)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The protected resources hosted on the resource server are the capabilities or functions of a receiver system called by an integration flow in the context of an integration scenario.</td>
</tr>
<tr>
<td>OAuth client</td>
<td>Party that wants to access the data of a resource owner that needs to be authorized.</td>
<td>The Cloud Integration runtime component that calls the receiver</td>
</tr>
</tbody>
</table>
When talking about Cloud Integration outbound communication towards a receiver system, protected resources are the capabilities of the receiver system addressed by the integration flow (outbound communication). The client in this picture is the Cloud Integration worker where the integration flow is deployed.

There are three grant types for OAuth 2.0. Grant types refer to the possible ways in which an application can get an access token.

- OAuth 2.0 Client Credentials
- OAuth 2.0 SAML Bearer Assertion
- OAuth 2.0 Authorization Code

A specific OAuth variant is used with the Cloud Integration Twitter and Facebook adapter.

### 8.1.1.2.3.1 OAuth 2.0 Client Credentials Grant

When OAuth 2.0 client credentials grant is implemented, the client gets access to the protected resources in two steps: After presenting a set of client credentials, the client fetches an access token from the token service. In a subsequent step, the client uses the access token to get access to the protected resources.
In detail, OAuth 2.0 client credentials grant is implemented in the following way for Cloud Integration outbound communication:

At **design time**, the integration developer performs the following tasks:

1. The integration developer requests the client credentials such as Client ID, Client Secret and token service URL from the administrator of the receiver system.
2. Once the receiver system administrator has shared the credentials and the URL, the integration developer specifies an **OAuth2 Client Credentials** artifact using information requested before and deploys the artifact on the Cloud Integration tenant. Likewise, the integration developer specifies the name of the **OAuth2 Client Credentials** artifact when configuring the receiver adapter of the integration flow involved (that is used to connect to the receiver system).

**i Note**

See: [Deploying an OAuth2 Client Credentials Artifact](#) [page 47]

After these steps have been executed successfully, at **runtime**, the authorization workflow is processed as outlined in the following chart:

1. Cloud Integration connects to the token service and presents the credentials.
2. The token service authenticates the client credentials and (if they are valid) provides an access token in return.
3. Cloud Integration authenticates itself against the receiver system (that hosts the protected resources in OAuth terms) with the help of the access token and requests access to the protected source.
4. The receiver system validates the access token and (if it’s valid) grants access to the protected resource.
8.1.1.2.3.2 OAuth 2.0 SAML Bearer Assertion Grant

When OAuth 2.0 SAML Bearer Assertion grant is implemented, the client (Cloud Integration) gets a SAML Bearer Assertion from a (SAML) custom identity provider. Using the SAML Bearer Assertion, the client requests an access token from a token service in a next step (using the SAML bearer assertion to proof its identity). Finally, the token service validates the SAML Bearer Assertion and passes back the OAuth access token that can be used to access the protected resources finally.

A SAML Bearer Assertion defines a user context that can be propagated between different systems in a communication scenario – a scenario known as *Principal Propagation*. A SAML Bearer Assertion contains a user and a public certificate that identifies the user at a custom identity provider. The SAML Bearer Assertion enables a component to request an access token from a resource server for the given user context.

The following receiver system types are currently supported:

- An SAP SuccessFactors system
- An SAP Cloud Platform system (either Neo or Cloud Foundry)

> Tip

For an example of how this grant type is used with an SAP SuccessFactors system, see:

- [OAuth SAML Bearer Assertion Flow Example](page 211) (maps the explanation of the grant type to a concrete system landscape and use case).
- [SAP Cloud Platform Integration – Principal Propagation with SuccessFactors OData V2](SAP Community blog describing step by step how to set up this example)

In detail, OAuth 2.0 SAML Bearer Assertion grant is implemented in the following way for Cloud Integration outbound communication:

At *design time*, the integration developer performs the following tasks:

1. The integration developer creates a trust relationship between the sender system and Cloud Integration.
2. The integration developer creates an OAuth client for Cloud Integration (required to define the connection from the sender to Cloud Integration).
3. The integration developer creates an OAuth client for the receiver system (required to define the connection from Cloud Integration to the receiver).
   
   You specify the signing certificate of the certificate defined when setting up the trust relationship (that way, you exchange the public certificate for the custom identity provider associated with the Cloud Integration tenant with the receiver system).
   
   During this step, a client key is created (that is needed to specify the security artifact for Cloud Integration). This is a key to access the API of the receiver system (API key).
4. The integration developer gets information such like token service URL, the type of receiver system, and additional information specific for the receiver system type. In case the SuccessFactors system type is chosen, a company ID is given that indicates the client instance used to connect to the SuccessFactors system. In case the SAP Cloud Platform (Neo or Cloud Foundry) system type is chosen, a token service user and password is given that indicates the user to access the token service.

   This information is needed to define the OAuth2 SAML Bearer Assertion artifact to be referred to in the related receiver adapter.
5. Once the information is known, the integration developer specifies an OAuth2 SAML Bearer Assertion artifact using information requested before and deploys the artifact on the Cloud Integration tenant.

Likewise, the integration developer specifies the name of the OAuth2 SAML Bearer Assertion artifact when
configuring the receiver adapter of the integration flow involved (that is used to connect to the receiver system).

More information: Deploying an OAuth2 SAML Bearer Assertion [page 49]

6. Set up the connection of the sender to Cloud Integration (for example, by defining a destination in the sender system). During this step, you need to specify token service URL, token service user, and password of the subaccount that hosts the Cloud Integration tenant.

When these steps have been executed successfully, at runtime, the authorization workflow is processed as outlined in the following chart:

1. The sender sends the client key to the SAML token issuer.
2. The SAML token issuer authenticates the client key and, if valid, provides the sender with the SAML Bearer Assertion (for the given user context).
3. The sender requests processing of the related integration flow on the Cloud Integration tenant (and provides the SAML Bearer Assertion with the request). This step is executed via an SAP Cloud Platform destination with OAuth2SAMLBearer authentication.
4. Cloud Integration connects to the token service and presents the SAML Bearer Assertion.
5. The token service validates the SAML Bearer Assertion and (if it is valid) provides an access token in return.
6. Cloud Integration authenticates itself against the receiver system with the help of access token and requests access to the protected resource. Note that the receiver system contains the protected resources in terms of OAuth.
7. The receiver system validates the access token and (if it’s valid) grants access to the protected resource.

**Related Information**

OAuth SAML Bearer Assertion Flow Example [page 211]
### 8.1.1.2.3.2.1 OAuth SAML Bearer Assertion Flow Example

In the example, OAuth SAML Bearer Assertion grant type is used with the SuccessFactors OData V2 receiver adapter.

You find a detailed description how to configure and set up this example step by step in the following SAP Community blog: [SAP Cloud Platform Integration – Principal Propagation with SuccessFactors OData V2](#).

We summarize how the components involved interact with each other, the steps how to configure the scenario, and the OAuth authorization workflow.

In this example, the user logs in to a sender app to fetch tasks retrieved from an SAP SuccessFactors system (which, in OAuth terms, contains the protected resources).

Cloud Integration is interconnected with the sender and SAP SuccessFactors. The connection to the SuccessFactors system is configured using the SuccessFactors OData V2 receiver adapter.

In this scenario, Cloud Integration fetches the tasks of the user logged in to the sender app. Therefore, the user context (principal) needs to be propagated from the sender app to Cloud Integration, and, finally, from Cloud Integration to SAP SuccessFactors. Principal propagation is achieved through the OAuth2 SAML Bearer assertion flow.

The sender app uses a custom identity provider that also acts as SAML token issuer.

The following figure maps the entities described generally in [OAuth 2.0 SAML Bearer Assertion Grant](#) to the concrete use case and system landscape given in the example.

When configured as described in the blog [SAP Cloud Platform Integration – Principal Propagation with SuccessFactors OData V2](#), the authorization flow works in the following way.

1. The user (logged in to the sender app deployed on account 1) invokes an action to get an SAP SuccessFactors entity (task).
2. The sender app requests the SAML assertion from the custom identity provider (providing the client key). The sender app communicates with the integration flow via an SAP Cloud Platform destination configured in account 1. When connecting to the custom identity provider to get the SAML assertion, account 1 communicates with the custom identity provider based on this destination.
3. The custom identity provider sends back the SAML assertion.
4. The sender calls the Cloud Integration endpoint (of the related integration flow).
   This step is executed via an SAP Cloud Platform destination with OAuth2SAMLBearer authentication.
5. Cloud Integration connects to the token service (part of SAP SuccessFactors) providing the information stored in the OAuth2 SAML Bearer Assertion credentials artifact (deployed on the Cloud Integration tenant and referred to in the SuccessFactors receiver adapter).
6. The token service returns the OAuth access token.
7. Cloud Integration uses the access token to request the SAP SuccessFactors entity.
8. SAP SuccessFactors checks if the token is valid and, if that's the case, returns the entity.
9. Cloud Integration returns the SAP SuccessFactors entity to the sender app.
10. The sender app returns the entity for the given user context.

### 8.1.1.2.3.3 OAuth 2.0 Authorization Code Grant

The grant type Authorization Code is the most complex grant type and offers an additional level of security. The user password is never visible to the OAuth client.

In detail, OAuth 2.0 Authorization Code grant is implemented in the following way for Cloud Integration outbound communication:

As a prerequisite to initiate the Authorization Code grant workflow, the integration developer performs the following tasks:

<table>
<thead>
<tr>
<th><img src="image.png" alt="Image" /></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>i Note</strong></td>
</tr>
<tr>
<td>The integration developer and the account user are typically the same person just with different roles.</td>
</tr>
</tbody>
</table>

1. The integration developer registers an application, so an OAuth 2.0 client (with client ID, client secret, authorization URL, and token service URL) is created. For the Mail adapter, the integration developer creates the OAuth 2.0 client in Microsoft Active Directory tenant.
2. The integration developer uses the Cloud Integration Web UI and creates an OAuth2 Authorization Code credentials artifact and deploys it on the Cloud Integration tenant. During this step, the integration developer specifies the parameters Client ID, Client Secret, Authorization URL, and Token Service URL based on the values generated when creating the OAuth client in the previous step. After this step, the credential is in status Unauthorized.
   See: Deploying an OAuth2 Authorization Code [page 51].

When these steps have been executed successfully, at the authorization workflow is processed as outlined in the following chart. Note that the workflow depicted in the figure comprises user actions (of the integration developer) and system steps (executed by the Cloud Integration worker).
The following chart explains the flow for OAuth 2.0 Authorization Code:

1. The integration developer authorizes the OAuth2 Authorization Code credentials artifact using the Cloud Integration Web UI.
2. Triggered by the Authorize action, Cloud Integration requests user authorization for certain scopes of the application from the token server.
3. The token service prompts a user login screen and requests the approval of the user for the app.
4. The account user grants approval.
5. After the user gave his/her approval, the token service returns the authorization code of the user to the Cloud Integration tenant.
6. Cloud Integration calls the OAuth 2.0 token endpoint of the token service with the client ID, client secret, and the authorization code.
7. The token service checks the request and sends a refresh token as response. It also sends an access token that is ignored by the Cloud Integration tenant.
8. Cloud Integration stores the refresh token and the user name together with the client Id, client secret, and the scopes in the OAuth 2.0 Authorization Code. The status of the OAuth2 Authorization Code credentials artifact changes to Deployed.
9. The Cloud Integration worker reads OAuth 2.0 Authorization Code information and calls the token service with client Id, client secret, and refresh token.
10. The token service sends back an access token.
11. The Cloud Integration worker connects to the receiver system protected by OAuth 2.0 and requests access with the access token and user name if necessary.

**Note**

- **Refresh Token**: A refresh token must be valid at least for 3 days, although Microsoft allows shorter validity periods (up to 10 minutes).
- **Refresh Token**: The refresh token is automatically updated before it expires by a scheduled job.
- **Max. Number of OAuth 2.0 Authorization Code Credentials**: 4500.
8.1.2.3.4 OAuth 2.0 Used With Twitter and Facebook Adapter

OAuth can be used for outbound communication with the Twitter or Facebook receiver adapter. In this case, the OAuth 2.0 roles are used in the following way:

- The tenant is the client that accesses Twitter or Facebook (as resource server).
- The Twitter or Facebook account owner is the user (that owns the protected resources which is Twitter or Facebook content).
- Using an API (for Twitter or Facebook), the user generates the OAuth 2.0 credentials (client credentials as well as token credentials) required in order to access the protected resources.
- The user provides the client (tenant) with the OAuth 2.0 credentials in the following way:
  - For each OAuth 2.0 credential, a separate Secure Parameter artifact is created and deployed on the tenant.
  - In the Twitter or Facebook adapter, the credential names are specified.

The following figure illustrates the OAuth 2.0 communication flow for this use case.

8.1.1.3 Load Balancer Root Certificates Supported by SAP

The load balancer supports a certain list of root certificates.

A system sending a message to the Cloud-based integration platform using HTTPS as secure transport channel is not directly connected to the tenant. Instead of this, a load balancer component is interconnected that terminates all inbound HTTPS requests, and re-establishes a new secure connection.

To set up a secure connection between a sender system and the integration platform, you therefore need to make sure that the sender system's keystore contains a client certificate that is signed by one of those certification authorities (CAs) that are trusted by the load balancer component of SAP.
The following list summarizes the root certificates that are supported by the load balancer.

**Note**

A specific certificate that identifies a certification authority (CA) is referred to as root certificate. Such a certificate is typically not signed by any other authority, as it is at the root of a certificate chain.

The load balancer component is owned by SAP, and you, the customer, don’t need to care how it is configured. However, you need to make sure that the client certificate in your sender keystore is signed by one CA that is listed below.

**Note**

Please be aware that only root certificates are being imported into the Keystore of the SAP Load Balancer! Therefore you as a customer must always assign the whole certificate chain to the certificate to enable the connected component to evaluate the chain of trust.

- Amazon Root CA 1
- Baltimore CyberTrust Root
- COMODO RSA Certification Authority
- Certum CA
- DigiCert High Assurance EV Root CA
- DigiCert Global Root CA
- DigiCert Global Root G2
- GeoTrust Global CA
- GeoTrust Primary Certification Authority - G3
- GlobalSign
- GlobalSign Root CA
- Go Daddy Class 2 Certification Authority
- Go Daddy Root Certificate Authority - G2
- Entrust.net Certification Authority (2048)
- Entrust Root Certification Authority
- Entrust Root Certification Authority - G2
- QuoVadis Root CA 2
- SAP Cloud Root CA
8.1.2 SFTP-Based Communication

8.1.2.1 How SFTP Works

A tenant can connect as SFTP client to an SFTP server (the latter either hosted at SAP or in the customer landscape).

Depending on the direction of data flow (whether the tenant reads data from the SFTP server or writes data to it), either an SFTP sender adapter or SFTP receiver adapter is involved.

Files are stored on the SFTP server in specific directories referred to as mailboxes. For each mailbox, a user is specified in order to control access to the data.

In certain cases, you have the option to choose between the following authentication options for SFTP connectivity in the SFTP (sender or receiver) adapter:

- User Name/Password
- Public Key

User Name/Password Authentication

The tenant connects to the server with a user and authenticates itself against the SFTP server with a password.

The user credentials (user name and password) are stored in a User Credentials artifact which has been deployed on the tenant prior to connection set up.
Public Key Authentication

In order to set up secure connection between the SFTP client and SFTP server, a combination of symmetric and asymmetric keys is applied.

- Symmetric (session) keys are used in order to encrypt and decrypt data within a data transfer session.
- Asymmetric key pairs (on client and server side) are used in order to encrypt and decrypt the session keys.

Symmetric and asymmetric keys are used by a client and a server exchanging data via SFTP in the following way:

1. The client connects to the server.
2. The server sends his public key to the client.
3. The client checks if the server is a trusted participant by evaluating a known_hosts file at client’s side: if the server’s public key is listed there-in, the identity of the server is confirmed.
4. The client generates a session key (to be used for one data transfer session).
5. The client encrypts the session key with the public key of the server.
6. The client sends the encrypted session key to the server. As public and private key of one party are mathematical correlated with each other, the server can decrypt the session key using its private key.
7. The session can now be continued in an encrypted way.
8. As part of the secure data transfer (using the session key exchanged by the step before), the client sends its public key to the server.
9. The server checks if the public key of the client is known to him (evaluating an authorized_keys file on the server side).
10. The server encrypts a random number with the client’s public key and sends it to the client.
11. The client decrypts the random number with its private key and sends the unencrypted random number back to the server. That way, the client authenticates itself on server side.

8.1.2.1.1 Inbound SFTP With Public Key Authentication

For an SFTP client connected to an SFTP server using the Public Key authentication option, the following artifacts have to be generated and stored at the locations summarized in the following table. The table also
shows which artifacts need to be exchanged between the client and the server (during the onboarding process):

<table>
<thead>
<tr>
<th>SFTP Client Side</th>
<th>SFTP Server Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public keys of all connected SFTP servers</td>
<td>Public keys of all connected SFTP clients on the SFTP server side</td>
</tr>
<tr>
<td>A public key is used in order to authenticate the SFTP server (as known host) on the SFTP client side. Public keys of all connected SFTP servers are stored in a <code>&lt;known_hosts&gt;</code> file on the client side.</td>
<td>This file has to be stored in an <code>&lt;authorized_keys&gt;</code> file on the SFTP server.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td>The <code>&lt;known_hosts&gt;</code> file contains the public keys and addresses of the trusted SFTP servers. The client checks if the server is a trusted participant by evaluating a <code>&lt;known_hosts&gt;</code> file on the client side: If the server's public key is listed there, the identity of the server is confirmed.</td>
<td>Generating this public key is the task of the expert that hosts the SFTP client.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td>Generating the public key of the SFTP server is the task of the expert that hosts the SFTP server.</td>
<td>Generating this public key is the task of the expert that hosts the SFTP server.</td>
</tr>
<tr>
<td>Private key of SFTP client (stored on client)</td>
<td>Private key of SFTP server (stored on server)</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td>The private key of the SFTP client can be either an RSA private key file or a DSA private key file. The private key (together with its associated public key) has to be stored in a keystore.</td>
<td>Generating this public key is the task of the expert that hosts the SFTP server.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td>Generating this private key is the task of the expert that hosts the SFTP client.</td>
<td>Generating this public key is the task of the expert that hosts the SFTP server.</td>
</tr>
</tbody>
</table>

A tenant can connect as an SFTP client to an SFTP server (the latter either hosted at SAP or in the customer landscape).

The following figure shows the basic setup of components used for SFTP for inbound communication (when the data flow is directed from an SFTP server to the tenant).
To specify the technical details of the message flow from the SFTP sender to the tenant (SFTP client), a sender SFTP adapter has to be configured for the related integration flow.

### 8.1.2.1.2 Outbound SFTP With Public Key Authentication

For an SFTP client connected to an SFTP server using the Public Key authentication option, the following artifacts have to be generated and stored at the locations summarized in the following table. The table also shows which artifacts need to be exchanged between the client and the server (during the onboarding process):

<table>
<thead>
<tr>
<th>SFTP Client Side</th>
<th>SFTP Server Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public keys of all connected SFTP servers</td>
<td>Public keys of all connected SFTP clients (used in order to authenticate the SFTP clients on the SFTP server side)</td>
</tr>
<tr>
<td>A public key is used in order to authenticate the SFTP server (as known host) on the SFTP client side.</td>
<td>This file has to be stored in an <code>&lt;authorized_keys&gt;</code> file on the SFTP server.</td>
</tr>
<tr>
<td>Public keys of all connected SFTP servers are stored in a <code>&lt;known_hosts&gt;</code> file on the client side.</td>
<td>Note Generating this public key is the task of the expert that hosts the SFTP client.</td>
</tr>
</tbody>
</table>

**i Note**

The `<known_hosts>` file contains the public keys and addresses of the trusted SFTP servers. The client checks if the server is a trusted participant by evaluating a `<known_hosts>` file on the client side: If the server’s public key is listed there, the identity of the server is confirmed.

**i Note**

Generating the public key of the SFTP server is the task of the expert that hosts the SFTP server.
A tenant can connect as an SFTP client to an SFTP server (the latter either hosted at SAP or in the customer landscape).

The following figure shows the basic setup of components used for SFTP for outbound communication (when the data flow is directed from the tenant to an SFTP server).

To specify the technical details of the message flow from the tenant (SFTP client) to the SFTP server, an SFTP receiver adapter has to be configured for the related integration flow.

### 8.1.3 Message-Level Security

Several standards are supported to protect the message content (message-level security).

Message-level security features allow you to digitally encrypt/decrypt or sign/verify a message (or both). The following standards and algorithms are supported.
### Message-Level Security Options

<table>
<thead>
<tr>
<th>Security Standard</th>
<th>Security Feature</th>
<th>Supported Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKCS#7/CMS Enveloped Data and Signed Data</td>
<td>Encryption/decryption of message content</td>
<td>Supported algorithms (by the symmetric key) for content encryption (format Cipher/Operation Mode/Padding Scheme): DESede/CBC/PKCS5Padding, DES/CBC/PKCS5Padding, AES/CBC/PKCS5Padding, ARFCOURE/ECB/NoPadding, Camellia/CBC/PKCS5Padding, RC2/CBC/PKCS5Padding, CAST5/CBC/PKCS5Padding.</td>
</tr>
<tr>
<td>PKCS#7/CMS provides a syntax for data that has cryptography applied to it, such as digital signatures or digital encryption.</td>
<td>Signing/verification of payload</td>
<td>Supported algorithms for content signing (digest and encryption algorithm): SHA512/RSA, SHA384/RSA, SHA256/RSA, SHA224/RSA, SHA/RSA, RIPEMD128/RSA, RIPEMD160/RSA, RIPEMD256/RSA, MD5/RSA, MD2/RSA, RIPEMD160andMGF1/RSA-ISO9796-2-2-3, SHAandMGF1/RSA-ISO9796-2-2-3, SHA256withDSA, SHA224withDSA, SHA/DSA.</td>
</tr>
<tr>
<td>The CMS specification can be found at: <a href="http://tools.ietf.org/html/rfc5652">http://tools.ietf.org/html/rfc5652</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digitally signing a message is based on the CMS type Signed Data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digitally encrypting or decrypting the content of a message is based on the CMS type Enveloped Data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKCS#7/CMS Enveloped Data and Signed Data</td>
<td>Encryption/decryption and signing/verification of payload</td>
<td>Supported algorithms (by the symmetric key) for content encryption (format Cipher/Operation Mode/Padding Scheme): DESede/CBC/PKCS5Padding, DES/CBC/PKCS5Padding, AES/CBC/PKCS5Padding, ARFCOURE/ECB/NoPadding, Camellia/CBC/PKCS5Padding, RC2/CBC/PKCS5Padding, CAST5/CBC/PKCS5Padding.</td>
</tr>
<tr>
<td>This is a subset of the algorithms that are supported for PKCS#7/CMS Enveloped Data and Signed Data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The generated signature does not conform to the CAdES-BES (CMS Advanced Electronic Signatures) signature standard.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Standard</td>
<td>Security Feature</td>
<td>Supported Algorithms</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Open Pretty Good Privacy (PGP)</td>
<td>Encryption/decryption of message content</td>
<td>Supported symmetric key algorithms for content encryption (symmetric key algorithms): CAST5 (128 bit key, as per [RFC2144]), Blowfish (128 bit key, 16 rounds), AES with 128, 192, and 256-bit key, Twofish with 256-bit key, DESede with 168-bit key. DES is not supported.</td>
</tr>
<tr>
<td></td>
<td>Encryption/decryption and signing/verification of the message</td>
<td>Supported signature algorithms for PGP signing: SHA-1, SHA224, SHA256, SHA384, SHA512, MD5, RIPE-MD/160</td>
</tr>
<tr>
<td>XML Signature</td>
<td>Signing/verification of payload</td>
<td>Supported signature algorithms: DSA/SHA1, RSA/SHA1, RSA/SHA256, RSA/SHA384, RSA/SHA512</td>
</tr>
<tr>
<td>XML Advanced Electronic Signature (XAdES)</td>
<td>Signing payload</td>
<td>The same signature algorithms as for XML Signature are supported.</td>
</tr>
<tr>
<td></td>
<td>Encryption/decryption of SOAP body</td>
<td>The default signature algorithm is set by the data in the certificate, that is, one of the following: <a href="http://www.w3.org/2000/09/xmldsig#rsa-sha1">http://www.w3.org/2000/09/xmldsig#rsa-sha1</a> or <a href="http://www.w3.org/2000/09/xmldsig#dsa-sha1">http://www.w3.org/2000/09/xmldsig#dsa-sha1</a>. The default signature digest algorithm is: <a href="http://www.w3.org/2000/09/xmldsig#sha1">http://www.w3.org/2000/09/xmldsig#sha1</a>.</td>
</tr>
</tbody>
</table>

Strong encryption is supported for the following algorithms:
- AES/CBC/PKCS5Padding
- Camellia/CBC/PKCS5Padding

For these algorithms, the key lengths 192 and 256 are possible.

**Recommendations**

Some algorithms (like MD2, MD5, DES or RC4) are still supported for legacy reasons, but they are not considered secure any more. We recommend that you check the official recommendations from National Institute of Standards and Technology (NIST) or European Union Agency for Network and Information Security (ENISA) for advice on algorithms and key strengths (for example, at: https://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-sizes-and-parameters-report).
8.1.3.1 How PKCS#7 Works

You have the option to digitally sign and encrypt message payloads based on PKCS#7/CMS Enveloped Data and Signed Data (PKCS stands for Public Key Cryptography Standards).

Signing and Verifying a Message

A digital signature ensures the authenticity of a message that way that it guarantees the identity of the signer and that the message was not altered after signing.

Digitally signing a message works the following way:

1. The sender calculates out of the message content a digest (or hash value) using a digest algorithm.
2. The sender encrypts the digest using a private key (type RSA or DSA). This is actually the signing step.
3. The sender sends the encrypted digest (which corresponds to the signature) together with the message content to the receiver.
4. The receiver decrypts the digest with the public key (which is related to the senders’ private key). The public key has the type RSA or DSA.
5. The receiver calculates the digest out of the content of the message (which has been sent to it by the sender).
   The receiver uses the same digest algorithm which the sender had used.
6. The receiver compares the decrypted digest (from the sender) with the one calculated at receiver side. In case both values (digests) are identical, the signature is verified.

i Note

The following algorithms for content signing are supported (digest and encryption algorithm):

PKCS#7 ensures that the digest algorithm is transferred together with the signature of the message and therefore available for the receiver.

This calculation is based on the message content. In case the message content has been transferred encrypted, a decryption step is needed before this step.
The following figure illustrates the process of digitally signing and verifying a message.

**Encrypting and Decrypting the Content of a Message**

Digital encryption allows you to encode the content of a message in such a way that only authorized parties can read it.

Digital encryption works two-stage based on symmetric and asymmetric key technology:

1. The sender encrypts the content of the message using a symmetric key.

   **Note**

   The following algorithms for content encryption (by the symmetric key) are supported (format Cipher/Operation Mode/Padding Scheme): DESede/CBC/PKCS5Padding, DES/CBC/PKCS5Padding, AES/CBC/PKCS5Padding, ARCFOUR/ECB/NoPadding, Camellia/CBC/PKCS5Padding, RC2/CBC/PKCS5Padding, CAST5/CBC/PKCS5Padding.

2. The sender encrypts the symmetric key using a public key.

   **Note**

   To encrypt the symmetric key, a public key of type RSA (with the cipher – or algorithm – RSA/ECB/PKCS1Padding) is used for each recipient.

3. The sender sends the encrypted message and the encrypted symmetric key to the receiver.
4. The receiver decrypts the symmetric key using a private key (which is related to the public key used by the sender).
For this decryption step a private key of type RSA is needed.

5. The receiver decrypts the content of the message using the decrypted symmetric key.

Strong encryption is supported for the following algorithms:

- AES/CBC/PKCS5Padding
- Camellia/CBC/PKCS5Padding

For these algorithms also the key lengths 192 and 256 are possible.

The following figure illustrates the process of digitally encrypting and decrypting the content of a message.

![Encryption and Decryption Process](image)

### 8.1.3.2 How XML Signature Works

A digital signature ensures the authenticity of a message that way that it guarantees the identity of the signer and that the message was not altered after signing. You have the option to digitally sign and validate a message based on the XML Signature standard (issued by the W3C consortium). Applying this standard means that the digital signature of a document itself is stored as an XML element.

XML Signature can be applied to any XML document.

The following options for XML Signature are supported:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enveloped Signature</td>
<td>Digital signature/validation is applied to XML element that contains the signature as an element (the Signature element). Using this option, the digital signature is part of the XML document to be signed/validated.</td>
</tr>
<tr>
<td>Enveloping Signature</td>
<td>Digital signature/validation is applied to content within an Object element which is part of the Signature element. That way, the Signature elements acts as an envelope for the signed content. Using this option, the digital signature is part of the XML document to be signed/validated.</td>
</tr>
</tbody>
</table>
You configure the usage of XML Signature in the related integration flow.

When applying XML Signature the following signature algorithms are supported: DSA/SHA1, RSA/SHA1, RSA/SHA256, RSA/SHA384, RSA/SHA512

When applying XML Signature the following canonicalization methods are supported: C14N, C14NwithComments, exc-C14N, exc-C14NwithComments

**Background Information**

In a simplified view, when configured correctly, digitally signing a message based on XML Signature implies the following main steps:

1. The sender of the message canonicalizes the XML message content that is to be signed. Canonicalization transforms the XML document to a standardized (reference) format. This step is required because an XML document can have more than one valid representations. Calculating a digest out of two different representations of the same document (according to step 2) results in different digests (or hash values). This would make the whole signing/validating process invalid.

2. Out of the canonicalized XML document, a digest is calculated using a digest algorithm.

3. The sender builds up a `SignedInfo` element that contains the digest.

4. The sender canonicalizes the `SignedInfo` element.

5. The sender builds a second digest for the `SignedInfo` element which contains the first digest.

6. The sender encrypts the digest using its private key.

7. The sender builds up the `SignatureValue` element which contains the encrypted digest from step 5 (the signature).

8. The message is sent to the receiver.

Digitally verifying (validating) a message based on XML Signature works the following way:

1. The receiver decrypts the encrypted digest (which is part of the `SignatureValue` element of the received message) using the sender’s public key.

2. The receiver calculates the digest out of the `SignedInfo` element of the message.

3. The receiver compares the two digests that result out of steps 1 and 2. That way it is the authenticity of the sender is checked.

4. The receiver canonicalizes the XML message content.

5. The receiver calculates the digest out of the XML message content.

6. The receiver compares the digest that results from the canonicalized message content with that one contained in the `SignedInfo` element of the message. That way, it is made sure that the content of the message has not been altered during message processing.

**8.1.3.3 How WS-Security Works**

Messages can be protected according to the WS-Security standard.

There are the following options:
• Digitally sign a message (and the other way round to verify a signed message)
• Digitally sign a message and to encrypt the message content (and the other way round to verify a message and to decrypt the message content)

**Note**


### 8.1.3.4 How OpenPGP Works

You can use Open Pretty Good Privacy (Open PGP) to digitally sign and encrypt messages.

OpenPGP gives you the following options to protect communication at message level:

- You can encrypt a payload.
- You can sign and encrypt a payload.

OpenPGP does not support signing without encryption or just verifying without decryption. The tenant expects either an encrypted payload or a signed and encrypted payload.

During runtime, the encryptor/.signer processor signs and encrypts the body of the inbound message and returns the resulting OpenPGP message in the body of the outbound message.

The required keys are stored in OpenPGP keyrings. The following types of keyrings exist:

<table>
<thead>
<tr>
<th>Type of Keyring</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGP secret keyring</td>
<td>Contains the public/private key pairs of the sender. It can contain multiple key pairs, each identified by a user ID. The private key is protected with a passphrase. For PGP secret keyrings deployed on tenants, the same passphrase has to be used to access all private keys of the PGP secret keyring.</td>
</tr>
<tr>
<td>PGP public keyring</td>
<td>Contains the public keys (related to the private keys that are stored in the PGP secret keyring of the communication partner).</td>
</tr>
</tbody>
</table>

**OpenPGP Signing/Verifying**

A digital signature ensures the authenticity of a message by guaranteeing the identity of the signer and that the message has not been altered since signing.

A message is digitally signed and verified as follows:

1. The sender calculates a digest (or hash value) from the message content using a digest algorithm. The following hash algorithms are supported for PGP signing:
   - For DSA key: SHA-1, SHA224, SHA256, SHA384, SHA512
   - For RSA key: MD5, SHA-1, RIPE-MD/160, SHA256, SHA384, SHA512, SHA224
2. The sender encrypts the digest using a private key (type RSA or DSA). This is the actual signing step.
The private key is looked up in the sender’s PGP secret keyring.

3. The encrypted hash value, together with the hash algorithm that has been used, is written to the signature element that is sent to the receiver together with the payload (as PGP signature format). The key ID of the signer of the private key is also written to the PGP signature format.

4. The receiver obtains the PGP signature format.

5. The receiver selects the key ID from the signature and uses the key ID to look up the right public key in the receiver’s PGP public keyring. This is the public key that corresponds to the private key used to sign the payload. In addition, the receiver checks whether the user ID (associated with the key ID) corresponds to an allowed user.

6. The receiver decrypts the hash value (and verifies the payload) using the public key.

The following figure illustrates the concept.

---

OpenPGP Encrypting/Decrypting

Digital encryption allows you to encode the content of a message in such a way that only authorized parties can read it.

A message is digitally encrypted and decrypted as follows:

1. The sender generates a symmetric key.
2. The sender encrypts the payload with the symmetric key.
   The following symmetric key algorithms for content encryption (symmetric key algorithms) are supported: TripleDES (168-bit key derived from 192), CAST5 (128-bit key, as per [RFC2144]), Blowfish (128-bit key, 16 rounds), AES with 128, 192, and 256-bit key. Twofish with 256-bit key DES is not supported.
3. The sender looks up a public PGP key in the PGP public keyring.
4. The sender encrypts the symmetric key using the public PGP key (from the PGP public keyring). You can use the following key types to encrypt the symmetric key: RSA and Elgamal (DAS is not supported).
5. The sender writes the encrypted symmetric key and the key ID into the Encryption Info element of the message. The key ID is used to identify the public key used for encryption (as the PGP public keyring can contain more than one public key). The Encryption Info element is sent to the receiver, together with the encrypted payload.

6. The receiver obtains the message and, based on the key ID (in the Encryption Info element), looks up the correct private key (associated with the public key used to encrypt the payload) in the PGP secret keyring. A passphrase is required to access the private key.

7. The receiver decrypts the symmetric key with the private key from the PGP secret keyring.

8. The receiver decrypts the payload with the symmetric key.

There is an option to compress data before the encryption step. The following compression algorithms are supported: ZIP [RFC1951], ZLIB [RFC1950], BZip2.

The following figure illustrates the concept.

The runtime supports the following features:

- Signing with several private keys (the resulting OpenPGP message then contains several signatures).
- Encryption with several public keys.
- Optional: OpenPGP compression and base 64 output or input.
- OpenPGP allows you to apply two different kinds of keys: primary keys and subkeys. (For simplicity, these are not differentiated in the figures above.) When you generate OpenPGP keys, a primary key with at least one subkey is created. Only the primary key can be used for certification, that is, to certify the trustworthiness of other keys. In addition, the primary key is also typically used to sign payloads. The subkey is used to encrypt payloads.

### OpenPGP Message Format Specification


<table>
<thead>
<tr>
<th>Packet Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Key Encrypted Session Key</td>
<td>Session key encrypted with a public key, key ID of the public key, and public-key algorithm</td>
</tr>
<tr>
<td>Packet Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Signature</strong></td>
<td>Binding between a public key and some data. There are several types of signature packets:</td>
</tr>
<tr>
<td></td>
<td>The certification, direct key, and subkey binding signature can be self-signed. The version 4 signature packet may also contain meta-information about the signature such as creation time, issuer, or key expiration time. The version 3 signature is deprecated.</td>
</tr>
<tr>
<td><strong>Symmetric Key Encrypted Session Key</strong></td>
<td>A symmetric key (also called session key) encrypted with a symmetric key; a symmetric algorithm is used. This packet is not supported.</td>
</tr>
<tr>
<td><strong>One-Pass Signature</strong></td>
<td>Placed at the beginning of the message before the data. It contains sufficient information to allow the system to start calculating the signature before the actual signature packet (which is after the data) is reached. There can be several such packets. One packet contains the public key algorithm, the hashing algorithm, the key ID of the signing key, and an indicator whether the signatures should be nested or not. A zero value indicates that the next packet is another One-Pass Signature packet that describes another signature to be applied to the same message data.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td>Nested signatures are not supported. However, several signatures over the same data in one PGP message are supported.</td>
</tr>
<tr>
<td><strong>Public Key</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Public Subkey</strong></td>
<td>Contains similar information to a public key package, but it denotes a subkey.</td>
</tr>
<tr>
<td><strong>Secret Key</strong></td>
<td>Contains all the information that is found in a public key packet, but also includes the secret key (encrypted private key).</td>
</tr>
<tr>
<td><strong>Secret Subkey</strong></td>
<td>Contains similar information to a secret key package, but it denotes a subkey.</td>
</tr>
<tr>
<td><strong>Compressed Data</strong></td>
<td>Typically, this packet contains the contents of an encrypted packet, or follows a Signature or One-Pass Signature packet, and it contains a literal data packet.</td>
</tr>
<tr>
<td>Packet Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Symmetrically Encrypted Data</td>
<td>Data encrypted with a symmetric key (using a symmetric key algorithm). The symmetric cipher used may be specified in a Public-Key or Symmetric-Key Encrypted Session Key packet that precedes the Symmetrically Encrypted Data packet. This packet uses a variant of the cipher feedback mode (CFB) (as defined at <a href="http://tools.ietf.org/html/rfc4880">http://tools.ietf.org/html/rfc4880</a>).</td>
</tr>
<tr>
<td>Literal Data</td>
<td>Contains plain data (binary or text).</td>
</tr>
<tr>
<td>User ID</td>
<td>Indicates the holder of a key. The package contains the user name, e-mail address, and comment of the keyholder.</td>
</tr>
<tr>
<td>User Attribute</td>
<td>Variant of the User ID packet, which can contain more information about the user. It is only used together with key material. This packet is not supported.</td>
</tr>
<tr>
<td>Sym. Encrypted and Integrity Protected Data</td>
<td>Variant of the Symmetrically Encrypted Data packet. It contains data that is encrypted with a symmetric key algorithm (using a symmetric key algorithm) and is protected against modification by the SHA-1 hash algorithm (less strong than a signature, but stronger than bare CFB encryption). It does not use Open PGP CFB mode but pure CFB mode.</td>
</tr>
</tbody>
</table>

**Restrictions for the Input Message Structure (for Decryptor/Verifier)**

The input payload must have the following packet sequence:

Public Key Encrypted Session Key ..., Sym. Encrypted and Integrity Protected Data | Sym. Encrypted Data, (Compressed Data,) (One Pass Signature ....,) Literal Data, (Signature ....)

Entries in brackets are optional, ellipses indicate repetition, commas represent sequential composition, and '|' separates alternatives.

For example, the Compressed Data packet is optional.

**Restrictions for the Output Message Structure (for Encryptor/Signer)**

The output PGP message is restricted to the following packet sequence:

Public Key Encrypted Session Key ..., Sym. Encrypted and Integrity Protected Data | Sym. Encrypted Data, Compressed Data, (One Pass Signature ....,) Literal Data, (Signature ....)

Entries in brackets are optional, ellipses indicate repetition, commas represent sequential composition, and '|' separates alternatives.
This does mean the following:

- A symmetric key cannot be encrypted with another symmetric key. The symmetric key that encrypts the payload cannot be encrypted by another symmetric key (which is, for example, generated from a password). OpenPGP allows this (see Symmetric Key Encrypted Session Key packet).
- Compression cannot be switched off. The Compressed Data packet is always mandatory. However, it is possible to choose the UNCOMPRESSED algorithm. In this case, the Compressed Data packet is still there, but contains the Literal Data uncompressed.
- Encryption is always mandatory. It is not possible to only sign data.
- Only one password for all private keys in the keyring can be used. This simplifies password maintenance.
- Nested signatures are not supported: If there are multiple signatures in the PGP message, they all contain the same hash value built over the original payload. OpenPGP does allow nested signatures where the enclosing signature is a signature of the enclosed PGP message including the enclosed signatures.
- DSA keys can only be combined with certain hash algorithms.

## 8.1.4 Certificate Management

Depending on the applied transport- and message-level security option, different types of security artifacts need to be managed and deployed on the tenant.

- X.509 certificates
  Used for transport-level security TLS and for message-level security using PKCS#7, WS-Security, and XML Digital Signature.
  They are stored in a Java keystore.
- PGP keys
  Used for message-level security using Open PGP.
- Known hosts files
  Required for transport-level security SFTP.
  SFTP keys are also stored in a Java keystore.

### Related Information

- X.509 Certificates [page 233]
- PGP Keys [page 237]
- Known Hosts File [page 237]
8.1.4.1 X.509 Certificates

X.509 certificates (that comply with the X.509 standard) are used for transport-level security TLS and for message-level security using PKCS#7, WS-Security, and XML Digital Signature.

Elements of X.509 Certificates

This topic does not explain the standard in detail, but points out the following important elements of an X.509 certificate.

A digital certificate provides a public key that is signed by a certification authority (CA).

Elements of X.509 Certificates

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer</td>
<td>Specifies the CA (that issued and signed the certificate).</td>
</tr>
<tr>
<td>Subject</td>
<td>Specifies the entity associated with the public key of the certificate.</td>
</tr>
<tr>
<td>Distinguished Name (DN)</td>
<td>Comprises the issuer, the subject, and other attributes.</td>
</tr>
<tr>
<td></td>
<td>A DN is a unique identifier of the certificate.</td>
</tr>
</tbody>
</table>

When you specify a certificate, you have to define additional attributes such as a company name, a country identification, and so on.

Related Information

Tenant Keystore [page 233]
Requirements for Keystore Passwords [page 237]
Certificate Chains [page 236]

8.1.4.1.1 Tenant Keystore

X.509 certificates and key pairs are stored in one keystore per tenant, referred to as tenant keystore.

Keystore Usage

A tenant keystore is used to secure message exchange both at transport level and at message level.
Transport-level security (HTTPS outbound connections from the SAP Cloud Platform Integration tenant to a remote system)

- Supporting client certificate authentication
  You can protect HTTP outbound connections by specifying client certificate authentication when configuring the related receiver adapter. If you do that, the receiver system authenticates the tenant (the client) based on a client certificate.
  To make this authentication option work, the tenant keystore needs to contain a client certificate which is a signed key pair containing a private and a public key.
  During the TLS handshake, one of the key pairs whose certificate chain is trusted by the server is selected for the TLS communication. If the server does not have a certificate of an appropriate certification authority (CA) in its trust store, the communication fails because the server cannot authenticate the client.
  If the server trusts several key pairs, one key pair is chosen at random for the connection.
  If you want to avoid random selection, you can specify an alias of a key pair entry in the related receiver adapter, so that only this specific key pair can be used in the TLS communication (use the Private Key Alias parameter for this purpose). If the keystore contains only one key pair or the server only trusts one key pair, this measure is not necessary. In some cases it is necessary to adapt the chain of the key pair. For example, if the chain of the key pair contains only the public certificate and the server contains only the root CA certificate, then you need to add the intermediate certificate to the chain of the key pair.
  More information: Client Certificate Authentication (Outbound) [page 205]

- Enabling the tenant to establish a trust relationship to the receiver system
  The SAP Cloud Platform Integration tenant also needs to establish a trust relationship to the receiver in such a way that the receiver can authenticate itself against SAP Cloud Platform Integration. In this case, authentication is accomplished based on a server certificate (as the receiver plays the role of a server). As prerequisite for this security measure, the tenant keystore needs to contain a (server) root certificate that is also trusted by the receiver.
  Even in case you specify basic authentication when configuring the related receiver adapter, you need to make sure that the tenant keystore contains a valid root certificate that is also trusted by the receiver.
  The keystore can also contain additional private keys with the aliases id_rsa or id_dsa (depending on the key type: RSA or DSA), which are used for the SFTP adapter.
  More information:
  Client Certificate Authentication (Outbound) [page 205]
  Basic Authentication [page 204]

Message-level security

The keystore also contains the public and private keys used for message-level security (signing and encryption). Public keys are used in the signature verification steps (XML Signature, PKCS#7/CMS Signature Verification, WebService Security) and in the encryption steps (PKCS#7/CMS, WebService Security) of integration flows. Private keys are used in the signature creation steps (XML Signature, PKCS#7/CMS Signature, WebService Security) and decryption steps (PKCS#7/CMS, WebService Security) of integration flows. In these steps, the relevant keystore entries are referenced by their aliases. We recommend using different keys for message- and transport-level security. Keep in mind that the expiration date of the certificates is not checked in the encryption/decryption steps and in the signing steps.
  More information: Message-Level Security [page 220]

Note that certain adapters (like the SOAP 1.x and the AS2 adapter) support options to sign/verify and encrypt/decrypt message content based on the Web Services Security (WS-Security) standard. To support such scenarios, the tenant keystore also needs to contain certain X.509 keys.
Keystore Content

There are the following entry types:

- **Key Pair entry**
  Consists of a private key and its X.509 certificate chain.
  All private keys of a keystore are encrypted with the same password. This password is also used as the
  keystore password (for checking the integrity of the keystore). The keystore is never stored in the same
database as the encrypted/signed application data. The password is stored in a separate database.
The certificate chain typically consists of the public key certificate and the intermediate certification authority (CA) certificate with which the signature of the public key certificate can be verified.

- **Certificate entry**
  In many cases this is an X.509 root certificate.

Keystore Management

A tenant keystore contains both entries owned by the tenant administrator (tenant owner) and entries owned
by SAP. SAP-owned entries cannot be changed or deleted by the tenant administrator and entries owned by the
tenant administrator cannot be changed or deleted by SAP.

More information: Managing Keystore Entries [page 56]

**i Note**

There is a dedicated naming convention for keystore aliases to indicate the owner of the keystore entry:
Alias names of SAP-owned entries start with `sap_` or are `hcicertificate`, `hcicertificate1`, `hcimsgcertificate`.

SAP Cloud Platform Integration does not verify the signatures of the certificates during the upload. Therefore,
the user who uploads the certificates is responsible for ensuring that the signatures of the certificates are
verified before the upload. Note that root certificates in particular must always be verified manually in any case.

Keystore Entries Preinstalled by SAP

When a customer gets a tenant provided by SAP, certain keystore entries have already been made available by
SAP.

- One **Key Pair entry** with the alias `sap_cloudintegrationcertificate`
• Certain Certificate entries which are also owned by SAP. These are root certificates that the customer can use to set up connections with other SAP cloud systems.

8.1.4.1.2 Certificate Chains

The trust relationship between a client and a server using TLS authentication is usually based on chain certificates.

When using the X.509 standard, a key pair used for the TLS handshake is usually signed by a certification authority (CA). This means that the server can assume that the public key (included in the certificate) provided by the client originates from a trusted source.

The X.509 standard allows you to build up hierarchical trust models. In such a model (also referred to as a certificate chain), many certification authorities (CAs) are involved on different hierarchy levels. This means that the certificate that identifies the CA as a trusted participant can itself be signed by a CA at a higher level in the hierarchy. This means that a number of (intermediate) CAs can be arranged above the actual client certificate. The highest level CA is called the root CA.

The following figure shows a certificate chain with two intermediate CAs:

![Certificate Chain Diagram]

We assume that the tenant is connected as a client to an external component (which can be referred to as the server or receiver system).

To establish SSL connectivity, the server is provided with the root CA certificate and nothing else. To make sure that a trust relationship between client and server can be established nevertheless, the client certificate (of the tenant) used for the SSL handshake has to contain the whole certificate chain. In other words, the client certificate has to include all intermediate CAs (excluding the root CA). This enables the server to evaluate and calculate the whole chain of trust.

Therefore, during connection setup (onboarding), the tenant key pair (client certificate) has to be assigned the whole certificate chain.

> Tip

To find out the certificate chain of the server, you can use the TLS Outbound Connection Test (accessible in the Monitoring application). This test also helps you to find out whether you have the correct CA certificate in the keystore to validate the server certificate chain (see option Validate Server Certificate of the Outbound Connection Test).

Related Information

TLS Connectivity Tests [page 97]
8.1.4.1.3 Requirements for Keystore Passwords

To protect a keystore, you have to specify a password when creating the keystore. You have to apply the following rules when specifying passwords for keystores:

- The password must have a minimum length of 8 characters.
- The password must contain characters of at least three of the following groups:
  - Lower-case Latin characters (a-z)
  - Upper-case Latin characters (A-Z)
  - Base 10 digits (0-9)
  - Non-alphabetic characters (!@#$%...)
- The password must not contain any characters from outside the standard ASCI table like, for example, German umlaut characters (äöü).

Example for password compliant with the above rule:

<xB+gku!kjhz>

8.1.4.2 PGP Keys

PGP public and secret keys (the latter containing a private key) can be uploaded to the tenant via separate keyrings. The PGP Public Keyring contains Transferable Public Keys as defined in section 11.1 of the Open PGP specification (https://tools.ietf.org/html/rfc4880) and the secret keyring contains Transferable Secret Keys as defined in section 11.2.

PGP keys are used in the PGP Encryptor and Decryptor step. You should only add PGP Public keys to the PGP Public Keyring if you trust this key. Typically you check the fingerprint of the public key. The same security measures must be taken for the secret keys which you use in the secret keyring. The encryption and signing steps do also work with expired certificates.

For the PGP Secret Keyring the same precautions as for the X.509 keystore must be taken because it contains private keys.

8.1.4.3 Known Hosts File

Known hosts files are relevant for SFTP communication. The known hosts file contains the host names and the public keys of the trusted SFTP servers. You should only have entries for those servers in the file which are used by the integration flows of the tenant and which you trust.
8.2 Security Elements

To set up the secure communication between a tenant and a sender/receiver system, certain security elements have to be created and - in some cases - exchanged between the involved components (the tenant on the one side and the sender/receiver system on the other side of the communication).

For example, to set up SSL communication using certificate-based authentication between a tenant and a receiver system, X.509 certificates are required. Those private keys owned by the tenant are to be part of a Java keystore that is to be deployed on the tenant, whereas the private keys owned by the receiver are to be part of the receiver system keystore. To complete the security setup, each keystore also has to contain the public key of the connected partner. In our example, the Java keystore of the tenant has to contain the receiver public key, and the receiver keystore has to contain the tenant public key.

This section provides a summary for each security option of how the required security elements have to be distributed among the involved components (tenant and sender/receiver systems).

Related Information

Security Elements (Transport-Level Security) [page 238]
Security Elements (Message-Level Security) [page 241]
How Security Artifacts Are Related to Integration Flow Configuration [page 245]

8.2.1 Security Elements (Transport-Level Security)

Each transport-level security option requires a specific set of security elements.

The following tables provide a summary of how the required security elements (in bold letters) have to be distributed among the involved components (tenant and sender/receiver systems).
<table>
<thead>
<tr>
<th>Security Option</th>
<th>Direction</th>
<th>Required by tenant administrator...</th>
<th>... to do the following</th>
<th>Required by sender/receiver administrator...</th>
<th>... to do the following</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS – basic authentication</td>
<td>Inbound (sender calls tenant)</td>
<td><strong>User name</strong> of SAP Cloud Platform (to be provided by sender administrator).&lt;br&gt;This is the user under which the customer system is to call the integration platform of SAP Cloud Platform.</td>
<td>Grant the required authorizations to enable this user to call the tenant.</td>
<td><strong>Load balancer root certificate</strong> (to be provided by tenant administrator).&lt;br&gt;Is required for the SSL communication step (can be obtained via the URL of the runtime node provided in the tenant mail by SAP).&lt;br&gt;Import into the keystore of the sender system.</td>
<td></td>
</tr>
<tr>
<td>Security Option</td>
<td>Direction</td>
<td>Required by tenant administrator…</td>
<td>… to do the following</td>
<td>Required by sender/receiver administrator…</td>
<td>… to do the following</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------</td>
<td>-----------------------------------</td>
<td>-----------------------</td>
<td>---------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>HTTPS – certificate-based</td>
<td>Inbound (sender calls tenant)</td>
<td><strong>Sender client root certificate</strong> (to be provided by sender administrator)</td>
<td>Check whether the CA the customer system used to get its client certificate signed is already part of the load balancer (server) keystore.</td>
<td><strong>Load balancer server root certificate</strong> (to be provided by tenant administrator)</td>
<td>Import into client PSE of the sender system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Sender client certificate</strong> (to be provided by sender administrator)</td>
<td>Configure the authorization check in the integration flow.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>List of trusted root certificates supported by load balancer</strong> (to be provided by tenant administrator)</td>
<td>Select a certification authority from the list for the certificate signing request for the client certificate.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Outbound (tenant calls receiver)</td>
<td><strong>Receiver server root certificate</strong> (to be provided by receiver administrator)</td>
<td>Import into tenant keystore (if not already there).</td>
<td><strong>Tenant client root certificate</strong> (to be provided by tenant administrator)</td>
<td>Import into the server PSE of the receiver system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tenant client certificate</strong> (to be provided by tenant administrator)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tenant public key</strong> (to be provided by tenant administrator)</td>
<td>Define the client certificate-to-user mapping for the configuration of authorization checks.</td>
<td><strong>SFTP server (receiver) public key</strong> (to be provided by SFTP server (receiver) administrator)</td>
<td>Add to known_hosts file (to be deployed as Known Hosts artifact on tenant).</td>
</tr>
<tr>
<td>SFTP</td>
<td>Outbound (tenant as SFTP client sends a request to an SFTP server)</td>
<td><strong>SFTP server (receiver) public key</strong> (to be provided by SFTP server (receiver) administrator)</td>
<td>Is required by tenant to check whether SFTP server can be trusted.</td>
<td><strong>Tenant public key</strong> (to be provided by tenant administrator)</td>
<td>Is used to authenticate tenant as a trusted SFTP client on the SFTP server side.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tenant public key</strong> (to be provided by tenant administrator)</td>
<td>Add to authorized_keys file on the SFTP server side.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 8.2.2 Security Elements (Message-Level Security)

The configuration of secure message exchange requires the exchange of public keys (or other security-related information) between the involved parties. Each message-level security option requires a specific set of keys to be exchanged.

The following tables provide a summary of how the required security elements (in **bold letters**) have to be distributed among the involved components (tenant and sender/receiver systems).

**Message-Level Security**

<table>
<thead>
<tr>
<th>Security Option/Standard</th>
<th>Direction</th>
<th>Protection Method on Tenant</th>
<th>Required by tenant administrator</th>
<th>... to do the following</th>
<th>Required by sender/receiver administrator</th>
<th>... to do the following</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKCS#7, WS-Security, XML Digital Signature (uses X. 509 certificates)</td>
<td>Inbound (sender calls tenant)</td>
<td>Decrypt</td>
<td>Tenant public key certificate (to be provided by tenant administrator)</td>
<td>Import into sender keystore</td>
<td>Tenant public key certificate (to be provided by sender administrator)</td>
<td>Is used to encrypt the message from the sender (that is to be encrypted by the tenant).</td>
</tr>
<tr>
<td>XML Digital Signature: only sign/encrypt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify</td>
<td>Sender public key certificate (to be provided by sender administrator)</td>
<td>Import into tenant keystore.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Encrypt</td>
<td>Receiver public key certificate (to be provided by receiver administrator)</td>
<td>Import into tenant keystore.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Option/Standard</td>
<td>Direction</td>
<td>Protection Method on Tenant</td>
<td>Required by tenant administrator … to do the following</td>
<td>Required by sender/receiver administrator … to do the following</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------</td>
<td>----------------------------</td>
<td>------------------------------------------------------</td>
<td>---------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sign</td>
<td></td>
<td>Tenant public key certificate (to be provided by tenant administrator)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Is used by the receiver to verify the message sent from the tenant.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OpenPGP (uses PGP keys) Inbound (sender calls tenant) Decrypt</td>
<td>Tenant public key (to be provided by tenant administrator)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Is used to encrypt the message from the sender (that is to be encrypted by the tenant).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>To make sure that the public key originates from the correct source and that it has not been changed on its way, consider the note below this table.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Option/Standard</td>
<td>Direction</td>
<td>Protection Method on Tenant</td>
<td>Required by tenant administrator …</td>
<td>… to do the following</td>
<td>Required by sender/receiver administrator …</td>
<td>… to do the following</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------</td>
<td>-----------------------------</td>
<td>-------------------------------------</td>
<td>-----------------------</td>
<td>---------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td></td>
<td>Verify</td>
<td></td>
<td>Sender public key (to be provided by sender administrator)</td>
<td>Import into tenant PGP public keyring.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Is used by the tenant to verify the signature of the message sent from the sender system.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To make sure that the public key originates from the correct source and that it has not been changed on its way, consider the note below this table.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outbound (tenant calls receiver)</td>
<td>Encrypt</td>
<td>Receiver public key (to be provided by receiver administrator)</td>
<td>Import into tenant PGP public keyring.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 8.2.3 Security Elements Related to the Mail Adapter

The usage of the mail adapter requires certificates both to validate the SSL connection and to encrypt the mail (in case S/MIME has been chosen).

The sender mail adapter enables the tenant to send an (encrypted) email to a receiver system, as illustrated in the following figure.

---

**Table: Security Option/Standard**

<table>
<thead>
<tr>
<th>Security Option/Standard</th>
<th>Protection Method on Tenant</th>
<th>Required by Tenant admin-istrator ... to do the following</th>
<th>Required by sender/receiver administrator ... to do the following</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign</td>
<td></td>
<td>Ten-ant public key (to be provided by tenant administrator)</td>
<td>Import into receiver PGP public keyring</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Is used by the receiver to verify the message sent from the tenant.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>To make sure that the public key originates from the correct source and that it has not been changed on its way, consider the note below this table.</td>
<td></td>
</tr>
</tbody>
</table>

---

**i Note**

Relevant for the SAP-managed operating model: When exchanging public PGP keys, note the following:

To ensure that the information originates from the correct source and that it has not been changed on its way, the key should be exchanged using a secure channel (for example, encrypted e-mail).

If a secure channel is not available, the person who receives the public key from the key owner has to verify the fingerprint of the public key. One option is to phone the owner of the public key and compare the fingerprint.
The tenant keystore needs to contain the following certificates:

### Certificates to be Imported into Tenant Keystore

<table>
<thead>
<tr>
<th>Certificate</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receiver server root certificate</td>
<td>For SSL connection: This certificate is required to identify the root CA that is at the top of the certificate chain that finally guarantees the trustability of the receiver server certificate.</td>
</tr>
<tr>
<td>Tenant client certificate</td>
<td>For SSL connection: This certificate is required to authenticate the tenant when calling the receiver system as client.</td>
</tr>
<tr>
<td>Public key (selected according to public key alias name configured in mail sender adapter)</td>
<td>This certificate is required to encrypt the email.</td>
</tr>
</tbody>
</table>

### 8.2.4 How Security Artifacts Are Related to Integration Flow Configuration

To specify the security-related aspects of the message flow, certain settings have to be made in the involved integration flows. These security settings are related to the security artifact deployed on the involved tenant.

The following example gives you an idea of how security artifacts and integration flow settings are related to each other: In order to specify in detail how a message is to be digitally encrypted, you need to define an Encryptor step in the relevant integration flow. At runtime, this Encryptor step needs to access the required public key to encrypt the message content. The public key itself has to be available in the keystore that is deployed on the involved tenant.

This section summarizes the following information for each security option:

- The required security artifact type (to be deployed on the tenant)
- The required step or adapter type (relevant for the related integration flow design)

#### Transport-Level Security Key Types

<table>
<thead>
<tr>
<th>Transport-Level Security</th>
<th>Key Type</th>
<th>Artifact Type (to Deploy on Tenant)</th>
<th>Integration Flow Step/Adapter Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS - Basic Authentication</td>
<td>User credentials (user name and password)</td>
<td>User Credentials</td>
<td>SOAP adapter, IDoc adapter, HTTP adapter, SuccessFactors adapter</td>
</tr>
</tbody>
</table>
### Transport-Level Security Key Types

<table>
<thead>
<tr>
<th>Key Type</th>
<th>Artifac Type (to Deploy on Tenant)</th>
<th>Integration Flow Step/Adapter Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS (SSL) - Certificate-Based Authentication</td>
<td>X.509 certificates</td>
<td>Keystore</td>
</tr>
<tr>
<td>SFTP (SSH)</td>
<td>SFTP key and known_hosts</td>
<td>Keystore SFTP adapter</td>
</tr>
</tbody>
</table>

#### Message-Level Security Key Types

<table>
<thead>
<tr>
<th>Message-Level Security</th>
<th>Key Type</th>
<th>Artifac Type (to Deploy on Tenant)</th>
<th>Integration Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKCS#7</td>
<td>X.509 certificates</td>
<td>Keystore</td>
<td>Signer, Encryptor, Decryptor</td>
</tr>
<tr>
<td>XML Digital Signature</td>
<td>This is the same key type as used when setting up HTTPS-based transport-level security.</td>
<td>Keystore</td>
<td>Signer, Encryptor, Decryptor</td>
</tr>
<tr>
<td>WS-Security</td>
<td>When setting up the security level for a tenant, you can use the same keystore for transport-level security keys (if you are setting up HTTPS-based communication) and for message-level security keys. Note, however, that we recommend using different keys for transport-level and message-level security.</td>
<td>Keystore</td>
<td>SOAP adapter (SOAP 1.x)</td>
</tr>
<tr>
<td>OpenPGP</td>
<td>PGP public keys and PGP secret keys</td>
<td>PGP public keyring, PGP secret keyring</td>
<td>Signer, Encryptor, Decryptor</td>
</tr>
</tbody>
</table>

The following figure illustrates the overall setup with regard to the tenant (for a situation where a keystore containing a public-private key pair is deployed on the tenant as a security artifact).
Deploy Keystore Security Artefact and Integration Flow

Sender → Tenant → Receiver
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.